University of A Coruña
Department of Computación

PhD thesis

A Logical Model of Information Retrieval based on Propositional Logic and Belief Revision

David Enrique Losada Carril

April 2001

University of A Coruña
Department of Computación

PhD thesis

A Logical Model of Information Retrieval based on Propositional Logic and Belief Revision

David Enrique Losada Carril

April 2001

UNIVERSIDADE DA CORUÑA

DEPARTAMENTO DE COMPUTACION

Facultade de Informática
Campus de Elviña, s/n.
15071 A Coruña
Telf. 981 167 000
Fax 981 167 160

Alvaro Barreiro García, Profesor Titular del Area de Ciencias de la Computación e Inteligencia Artificial en la Universidad de A Coruña.

CERTIFICA

Que la memoria titulada "A logical model of Information Retrieval based on Propositional Logic and Belief Revision" ha sido realizada bajo mi dirección por *David Enrique Losada Carril* en el Departamento de Computación de la Universidad de A Coruña, y concluye la tesis que presenta para optar al grado de Doctor en Informática.

A Coruña, 28 de Marzo de 2001

Alvaro Barreiro García
Director de la tesis

Vº Bº Jose Luis Freire Nistal
Director del Departamento de Computación

Universidade da Coruña · DEPARTAMENTO DE COMPUTACION

Contents

Acknowledgments

First of all I would like to thank my supervisor, Alvaro Barreiro for his invaluable support during these years. He introduced me to academic research and he brought me into the field of Information Retrieval. I want to thank him for giving me the opportunity of writing a thesis under his supervision. All the material presented here came out after useful discussions with him. Actually we did collaborate as colleagues and not as student-supervisor. Besides, he provided me with such an excellent environment for research as the Artificial Intelligence Laboratory (AILab) at the Department of Computer Science of the University of A Coruña.

Ramón P. Otero, who is the AILab leader, had also an important impact on this dissertation. He has always been open to discuss topics related to my research. I also thank him for encouraging me to keep myself close to the *AI-side* of this thesis. This was especially helpful when I was doing evaluation and, thus, I was overwhelmed by ugly numbers. Thanks to all the other members of the AILab, who have always been willing to help me with my dissertation. Particularly, some of the *Friday meetings* focused on my thesis and, thus, I could see my research from an AI point of view. These brainstormings were really very helpful. Specially, Pedro Cabalar and Raúl Ramos supported me with my research in many ways.

I am also very grateful to Professor C. J. van Rijsbergen who gave me the opportunity of visiting his research group at the University of Glasgow. I visited Glasgow for three months in the spring of 2000. My visit was really very fruitful for my work. I could get feedback from a research group that has led the field of Information Retrieval for many years. I thank Professor van Rijsbergen, Iadh Ounis and Iain Ruthven who discussed with me some topics related to my thesis. All the people from the Glasgow IR group were very familiar with me. I really appreciate how they adopted me in their group. They made me feel as a member of their group. Mirna Adriani, Di Cai, Gianni Amati, Joemon Jose, Robert Villa and Juliet van Rijsbergen also helped me during my stay at Glasgow. I would also like to thank my office mates, Marcos Theophylactou and Tassos Tombros, who introduced me in the social life of Glasgow.

I want to thank many other people that at one point or another provided useful input on topics related to the dissertation. I thank Ricardo Baeza-Yates, who visited our department in the summer of 1999. He gave me wise advises for presenting my paper at my first SIGIR conference, SIGIR'99. I also thank Mark Sanderson, who invited me to visit the University of Sheffield in May 2000. Mark discussed with me several aspects of my research and, thus, I could see my work from a point of view of a researcher from the classical IR community. In September 2000 I visited Mounia Lalmas at the University of London. She strongly encouraged me to continue my research on logical modeling for IR and I thank her comments and suggestions about my dissertation.

Fabio Crestani was also very helpful in my research. I could visit him at the University of Strathclyde during my stay at Glasgow. He made several important suggestions and he gave me new ideas to try out. Besides, he visited our university in September 2000 and we could

meet him to have further discussions. These meetings were very productive. Some of the work presented here has been inspired by him.

I want to thank the members of my reading committee for reviewing this thesis: Prof. Dr. José Mira, Prof. Dr. José L. Freire, Dr. Ramón P. Otero, Dr. Fabio Crestani and Dr. Mounia Lalmas.

I would also like to thank the financial support supplied by several organizations. I acknowledge the *Ministerio de Educación y Cultura* for the FPI grant that allowed me to develop this research work during these years. I also thank the *Ministerio de Educación y Cultura* for the stay grant without which my visit at Glasgow for three months might never have taken place. I also thank the University of Coruña for several travel grants that allowed me to travel and present research in progress all around the world. I thank the Association of Computing Machinery (ACM) which supported my travel to SIGIR'99 through a student travel grant. I also received finantial support from projects ref. XUGAPGIDT99PXI10201B (from the *Xunta de Galicia*) and ref. PB97-0228 (from the *Ministerio de Educación y Cultura*), which gave me the possibility to participate in some research conferences. The *Xunta de Galicia* also gave me support through a grant during my first year at the laboratory.

Furthermore my thanks go to my old mates Mario Otero and José M. Rodríguez. They introduced me in the AIlab some years ago and they become very loyal friends of mine. Now they are not at the lab and I have to say that I really miss them.

I would also like to thank my parents, who have always minded my work and supported my decisions. Last but not least, I profoundly thank Maria for her continuous support during all these years. I think it is not easy to bear an IR researcher, especially during evaluation stages. She has always encouraged me to do research and this work could not have been finished without her. I dedicate this thesis to her.

Chapter 1

Motivation and outline

Information Retrieval (IR) is the science concerned with the effective and efficient retrieval of information for the subsequent use by interested parties. Hence, IR systems have to deal with representation, storage, organization of, and access to information items. For centuries, man has organized, retrieved and used information. Originally, manual indexes were created to provide faster access to the information. For instance, librarians have been using them for a rapid access to the data for many years. More recently, the advent of computer systems gave IR a new insight. Nowadays, research in IR comprises a great variety of topics including modeling, document classification and categorization, systems architecture, user interfaces, data visualization, filtering, languages, etc. Besides, the enormous success of the Web in the nineties has attracted renewed interest in IR. The Web poses new problems and IR techniques arise as promising solutions.

Let us consider a store of documents and a person who has an information need. The basic problem in IR is the quest to find the set of documents which satisfy the user's information need. A document belonging to the document base is *relevant* if it supplies information which is useful to the user for satisfying his/her need (i.e. relevance is user-dependent). The search for relevant documents is expected to be done automatically by a system. This implies the definition of a method to store machine-readable representations for the elements involved in the problem. Information needs are usually expressed as queries in a query language supported by the system and documents are often represented in indexing structures. The concrete document's representation depends on the characteristics of the index. This includes natural language, sets of terms and many others. Usually, there is a significant gap between the representations of documents and queries and the documents and information needs themselves. A query is often a very awkward approximation to the information need that the user has in mind. Besides, to characterize the contents of a document is a fundamental problem. Even using the natural language text of a document, many questions remain unsolved. In this respect, there is a big question: *What is a document actually about?*. The quest for the answer of this question is a tremendous challenge in IR.

Another difficulty comes from the notion of *relevance* itself. Even having an appropriate method to extract the important information from documents and information needs, we have to know how to use this information to decide relevance (system relevance). IR systems often assume that relevance is binary, i.e. a document is relevant or is non-relevant. Besides, relevance is usually assumed to be fixed over time. Clearly, this is a simplification because a) a given document can be much more relevant to the user than other relevant documents and b) a document can be relevant to a user at a given moment but it can be considered non-relevant later (i.e. novelty of information plays a role in IR as well). Another problem is that relevance

is user-dependent and, hence, subjective. As a consequence, IR systems have to face up to the difficulties due to the subjectiveness of the IR process.

High speed computers with huge amounts of storage have solved most of the mundane tasks in IR, such as cataloging, administration, fast access, storage of huge amount of data and so forth. Nevertheless, the problem of effective retrieval remains largely unsolved. As argued in the previous paragraph, there are fundamental questions whose solution is still to come. The basic IR problem is intrinsically hard and current models do the best they can in order to get close to the solution. However, many of the assumptions adopted are too strict leading to IR models which are self-limited.

The representations used by classical IR systems exemplificate these limitations. Index terms (often named keywords) are usually adopted to index and retrieve documents. We can think in terms as words appearing within the text of a document. Nevertheless, not all the terms that a document mentions are equally significant to decide the contents of the document. Prepositions, adverbs, conjunctions, etc. are often eliminated (this is what is called stopword processing) and words are usually reduced to their syntactical root (this is what is called stemming). However these preprocessing techniques are just a first step to obtain a good set of terms that characterizes a document. Many nouns, adjetives, etc. are also meaningless to determine the actual topics of a document. As a consequence, many techniques have evolved to extract good terms or keywords that characterize the actual contents of a document. Statistical methods have played an important role in this issue for many years. It is usual that the importance of a term in a document is measured depending on factors such as the frequency of appearance of the term within the text of the document, the frequency of appearance of the term within the whole document collection and other variants. Of course, to index documents and queries through flats structures of terms is an oversimplification because a lot of the semantics in a document or user request is lost when we replace its text with a set of words. Unfortunately all the attempts which have tried to capture the semantics of the language in a more general way have failed or only worked in very restricted environments.

IR systems are based on a model within which relevance decisions can be quantified. Formally, a retrieval model may be defined as a set of three main components, namely:

- A model for the documents.
- A model for the queries.
- A matching function which defines the way a query is matched against any modeled document, i.e. the matching function implements the notion of system relevance.

This allows to present the user with a set of documents, which is the answer of the system to his/her information need. The ultimate goal of an automatic retrieval strategy is to retrieve all the relevant documents and to retrieve as few non-relevant documents as possible.

There is a great variety of models of IR depending on how they formalize the previous three elements. In the next section we will present the basic notions of two classical models, namely, the boolean model and the vector space model. Our objective is that the reader becomes familiar with the IR process and, hence we depict two basic models whose mechanisms are easy to grasp. Later, we will cover logical models of IR and we will motivate our decision of proposing a logical model for IR.

1.1 Classical Models

The purpose of this section is to cover two of the most important IR models proposed over the years. Distinct models adopt distinct ways to formalize the main actors around the retrieval process. We have chosen two well-known classical models as illustrative examples of the field. The boolean model is considered as *set-theoretic* because it represents documents and queries as sets of terms. On the other hand, the vector space model is considered as *algebraic* because it represents documents and queries as vectors in a t-dimensional space. The next paragraph presents some definitions used in the rest of the section.

Given an index term k_i and a document d_j, $w_{ij} \geq 0$ is the weight associated with the pair (k_i, d_j). This weight quantifies the importance of the index term k_i for describing the semantic content of the document d_j. The function g_i returns the weight associated with the index term k_i in any t-dimensional vector, i.e. $g_i(\vec{d_j}) = w_{ij}$.

1.1.1 Boolean Model

The boolean model is based on set theory and boolean algebra. It considers that index terms or keywords are either present or absent in a document. Documents can be regarded as binary-weighted vectors, i.e. the index term weights are assumed to be all binary: $\forall i, j \quad w_{ij} \in \{0, 1\}$. Queries are composed of index terms linked by the boolean connectives `AND`, `OR`, `NOT`. The boolean model's retrieval strategy is based on a binary decision criterion, i.e. each document is either relevant or non-relevant. There is no notion of a partial match to the query conditions.

The process of computation of the similarity between documents and queries is as follows. Since queries are conventional boolean expressions, they can be represented as sets of *conjunctive vectors*. For instance, the query q =`NOT` k_a `AND` (k_b `OR` k_c) can be written as $q_C = \{(0,1,0), (0,0,1), (0,1,1)\}$, where each element of the set is a binary-weighted vector whose first component is associated with the index term k_a, the second component is associated with k_b and the third component is associated with k_c. Each binary-weighted vector is called a conjunctive component of the query. Let q be a query and let q_C be the query transformed into a set of conjunctive vectors, the similarity of a document d_j to the query q is defined as:

$$sim(d_j, q) = \begin{cases} 1 & \text{if } \exists \vec{v} \in q_C | \forall k_i, g_i(\vec{d_j}) = g_i(\vec{v}) \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

The document d_j is considered relevant to the query q iff $sim(d_j, q) = 1$. Otherwise, the document is predicted to be non-relevant.

Since boolean queries have precise semantics and the concept of a set is quite intuitive, the boolean model provides a simple and clean framework which is easy to grasp by a common user of an IR system. Besides, since documents are just bags of terms, the matching process is very fast. Nevertheless, the boolean model suffers from some drawbacks. First, retrieval performance is limited because the relevance decision is binary. This may lead to retrieval of too few or too many documents. Moreover, it is not easy to translate an information need into a boolean expression. Most users find it difficult to express their requests through boolean expressions. Often, boolean queries are awkward representations of user's needs. Despite these drawbacks, the boolean model was adopted by many of the early commercial bibliographic systems and, nowadays, it keeps being the dominant model in the field of document database systems.

1.1.2 Vector Space Model

Despite the simplicity of the boolean model, it is well known that the use of non-binary weights leads to substantial improvements in retrieval performance. In this line, the vector space model assigns non-binary weights to terms in both documents and queries and it provides a framework in which partial matching is possible.

Documents and queries are represented as t-dimensional vectors as follows. Documents are represented as vectors $\vec{d} = (w_{1j}, w_{2j}, \ldots, w_{tj})$ and queries are represented as vectors $\vec{q} = (w_{1q}, w_{2q}, \ldots, w_{tq})$, where each weight w_{ij}, w_{iq} is positive and non-binary. The term weights are used to compute the degree of similarity between each document stored in the document base and the query supplied by the user. Documents are sorted in decreasing order of similarity and, thus, the vector space model takes into consideration documents which match the query terms only partially. This implies that the ranked answer set is a lot more precise than answer sets supplied by the boolean model.

Since documents and queries are vectors in a dimensional space, measures of correlation between vectors can be applied to get a measure of similarity between documents and queries. One of the most widely used measures computes similarity through the cosine of the angle between the two vectors. Formally,

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{ij} \times w_{iq}}{\sqrt{(\sum_{i=1}^{t} w_{ij}^2)} \times \sqrt{(\sum_{i=1}^{t} w_{iq}^2)}}, \tag{1.2}$$

where $\bullet$ stands for the inner product between two vectors.

The value of $sim(d_j, q)$ varies from 0 to 1 and a document might be retrieved even if it matches the query only partially. Many other matching functions have been designed to compute the degree of similarity between a query vector and a document vector. For a detailed study of them we refer to [3].

Usually, a threshold is established and the retrieved documents are those ones with a degree of similarity above the threshold. The user is presented with a ranked list of documents and, thus, he/she can inspect the list from the top document, i.e. the one with the highest degree of similarity.

Index term weights can be computed in many different ways. Distinct approaches apply distinct intuitions in order to determine which are the circumstances that make that a term is important. The tf/idf weighting scheme is one of the most popular methods to assign weights for terms in documents. Two main intuitions stand behind the tf/idf approach. First, terms appearing frequently within a document are considered as good representatives for the document and, thus, they should receive a high weight. Second, terms which appear in many documents are not very useful for distinguishing a relevant document from a non-relevant one and, thus, they should receive a low weight. Formally, let N be the size of the document collection and n_i be the number of documents in which the keyword k_i appears. Let $freq_{i,j}$ be the raw frequency of the term k_i in the document d_j, i.e. the number of times the term k_i is mentioned in the text of the document d_j. Then, the term-frequency factor, *tf-factor*, is given by the normalized frequency $f_{i,j}$ of the keyword k_i in document d_j. Formally,

$$f_{i,j} = \frac{freq_{i,j}}{max_l freq_{l,j}}, \tag{1.3}$$

where the maximum is computed over all terms which are mentioned in the text of the document d_j. Then, the *idf-factor* (idf stands for inverse document frequency) is given by:

$$idf_i = log\frac{N}{n_i} \tag{1.4}$$

Finally, the tf/idf weighting scheme assigns the following weight to the term k_i in the document d_j.

$$w_{i,j} = f_{i,j} \times idf_i \tag{1.5}$$

For query term weights a similar procedure is followed. Several variations of the above weighting scheme have been presented in the literature. However, in general, the tf/idf scheme provides a good method to assign weights for terms in many collections.

From a theoretical point of view, the vector model has the disadvantage that index terms are assummed to be mutually independent. Although this is an oversimplification, to consider term dependencies might hurt the overall retrieval performance. The vector model yields ranked answer sets which are difficult to improve upon without query expansion or relevance feedback. In general, the vector model is considered to be either superior or almost as good as the known alternative models.

1.1.3 Alternative models

Over the years, many other models have been proposed. The probabilistic model is based on the use of probability theory for modeling documents and queries. Given a user query, there is an ideal set of documents which contains exactly the relevant documents and no other. The query process is considered as a process of specifying the properties of that ideal answer set. The initial guess allows us to generate a preliminary probabilistic description of the ideal answer set which is used to retrieve a first set of documents. Interaction with the user is then initiated with the purpose of improving the probabilistic description of the ideal answer set.

Fuzzy and extended boolean models are alternative set-theoretic models. The fuzzy set model considers that each query term defines a *fuzzy* set and that each document has a *degree of membership* in this set. Extended boolean models extend the boolean model with the functionality of partial matching and term weighting.

The generalized vector space model is a variation of the vector space model in which index terms vectors are assumed linearly independent but are not pairwise orthogonal. The latent semantic indexing model maps each document and query vector into a lower dimensional space which is associated to concepts. The index term vectors are mapped into this lower dimensional space. Retrieval in the reduced space is expected to be superior to retrieval in the original space of index terms.

The inference network and the belief network models are variations of the probabilistic model. The inference network model associates random variables with index terms, documents and user queries. Index term and document variables are represented as nodes in a network and edges are directed from a document node to its term nodes to indicate that observation of the document yields improved belief on its term nodes. The query variable is also represented by a node in the network. Edges are directed from index term nodes to the query node. The belief network model is a variation of the inference network model that adopts a clearly defined sample space.

Unlike the inference network model, the belief network model's topology provides a separation between the document and the query portions of the network.

Our aim is not to cover the details of all the models proposed in the literature and we refer to the book by Baeza-Yates and Ribeiro-Neto [3] for a complete description of every important model of IR.

1.2 Why do we propose a logical model?

The time spent by indexing and retrieval tasks is a critical factor in IR systems. Huge amounts of data should be automatically indexed and, given a user query, the response time is fundamental for the success of a retrieval engine. This makes that classical approaches are mainly guided by efficiency rather than expressiveness. For instance, the boolean model and the vector space model retrieve efficiently documents but the expressiveness of the representations of documents and queries is poor. In general, there have been few attempts to increase the expressiveness of the representations of queries and documents.

The Artificial Intelligence (AI) research community has largely studied the tradeoff between the expressiveness of the representations and the efficiency of the reasoning tasks. Usually the proposal of a new Knowledge Representation (KR) language includes a detailed study of the complexity of the associated tasks. Depending on the constructors that we use to build the expressions of the language we can get a different level of complexity. Furthermore the area of KR has studied the problem of knowledge representation for many years. The IR problem connects directly with the field of KR. Indeed, documents are sources of knowledge and they should be treated as such. KR tools are adequate for representing the elements involved in the IR process. On the contrary, classical approaches tend to take simplistic positions. For instance, a vector is not appropriate for representing structured documents, such as research articles with several sections. The expressiveness of KR languages allows us to construct more general representations. Besides, there are specialized logics which are oriented to certain uses of knowledge.

When proposing a model for IR we have to face two main points regarding expressiveness. First, we have to assure that the retrieval process is efficient. Then, given the representations of a document and a query, the matching process has to be fast. In this sense, we should clearly establish a boundary for the expressiveness of the representations that guarantees an efficient matching process. AI's experience on this sort of tradeoffs can be very helpful. Moreover, the special characteristics of the IR environment introduces additional considerations. Document's representations should be obtained in an automatic way. This imposes an additional tradeoff because once an expressive framework is proposed, we have to specify methods that generate automatically the representations of the documents.

The known limitations of the classical IR models have caused researchers to propose new models from time to time. The need for designing and implementing more powerful retrieval models is by now more urgent than ever. Evolving from unstructured textual information, there is now a great variety of document's types including structured documents, multimedia documents and so forth. This evolution is challenging for IR and future models will have to deal with this extended notion of document. In this sense, we strongly believe that logic-based approaches are promising. The semantics of logic gives us the ability to write document's representations which capture the contents of documents in a better way. Besides, logic can formally encompass semantic properties of domain and expert knowledge. Unfortunately, all attempts towards the

explicit processing of knowledge have been mainly informal, or limited to retrieval systems for concrete domains.

Logic provides us with a rich and uniform framework in which information items can be modeled. Indeed, some logical models are able to embody within an homogeneous framework distinct features of IR systems, such as hypermedia links, multimedia content, user's knowledge and intentions and the nature of IR agents. Some of these features have been represented in other IR models but, often through *ad hoc* extensions. Furthermore, logic makes it possible to reason about an IR model and its properties. This is becoming increasingly important since conventional evaluation procedures, although good indicators of the retrieval performance of IR systems, often produce results which cannot be predicted or results which are hardly explainable. As a matter of fact, investigations are being carried out into various logic-based frameworks in order to get a theory which allows to predict the behavior of IR systems, compare them and prove properties about them. Although this is a great challenge, this would lead to a unified information-based model theory for expressing the semantics of IR. With such a meta-model, it would be possible to reason about various IR features of the model. This possibility is becoming increasingly important because conventional evaluation methods such as precision and recall often cannot be used on alone to study the performance of advanced IR systems.

In the next section we present a brief introduction to some logical models of IR. We explain the pioneering proposals and the basic notions of some logical approaches. This pretends to illustrate how logicians face the IR process.

1.3 Logical Models

The basic point of a logical model for IR is the assumption that queries and documents can be represented effectively by logical formulas. The retrieval process is based on some form of inference provided by the involved logic, i.e. logical approaches model IR as an inference process. The basic logical test is to decide whether or not the formula representing a query can be inferred from the formula representing a document.

Foundational approaches were directed to the use of classical logic. Documents and queries are represented by classical logic formulas and the notion of logical consequence is utilized to decide relevance, i.e. a document d is relevant to a query q iff $d \models q$, where $\models$ is the classical entailment, which consists on evaluating whether or not every model of d is a model of q. However, classical logic is not enough to model the IR process. It is necessary to take into account the uncertainty inherent in such an implication. Clearly, not all the documents that fail to entail (in the classical way) a given query are non-relevant. Some of them can deal with most of (but not all) the query topics and, thus, they are likely significant to the user. Hence, it is desirable to have a method to quantify the degree of *satisfaction* of $d \models q$. Unfortunately, classical logic does not allow to distinguish between documents that do not entail the query. This is because $\models$ provides us with a rigid test in which partial relevance cannot be modeled.

C. J. van Rijsbergen defined the basis for any logical approach that tries to capture the uncertainty of an implication with the form $d \rightarrow q$ [61] (the symbol $\rightarrow$ stands for the implication of the involved logic). Van Rijsbergen made an explicit connection between non-classical logics and IR modeling through the following *logical uncertainty principle*:

> Given any two sentences x and y; a measure of the uncertainty of $y \rightarrow x$ relative to a given data set is determined by the minimal extent to which we have to add information to the data set, to establish the truth of $y \rightarrow x$.

In this way, the main guidelines for logical approaches of IR were formally established. Since this principle does not suggest which logic to use, various logics and uncertainty theories have been proposed and investigated in the literature. The choice of the appropriate logic and uncertainty mechanisms has been a main research theme in logical IR modeling leading to a number of different approaches over the years. Distinct strategies implement van Rijsbergen's logical uncertainty principle in a different way. The book [12] is a detailed study of the work developed in the area. Lalmas and Bruza [34] defined a conceptual framework in which several logical approaches were analyzed. Specifically, a set of conceptual components of the logical uncertainty principle is used to study the behavior of different logical approaches.

1.3.1 Some logic-based approaches

Nie [46, 47] proposed the use of modal logic in order to get a more general model for IR. Modal logic introduces a possible-world semantics in which there is a set of worlds which are connected to each other via an accessibility relation. Each formula is evaluated within the context of a possible world. In modal logic the fact that a formula p is true with respect to a world w is written $M \models p[w]$. Two modal operators are introduced, namely the possibility $\Diamond$ and the necessity operator $\Box$. A formula $\Diamond p$ is true in a possible world w if p is true in at least one world accessible from w. A formula $\Box p$ is true in a possible world w if p is true in all the worlds accessible from w. Nie found an interesting connection between the semantics of the possibility operator and the evaluation of uncertainty needed for IR. Each document is represented by a set of logical formulas. This set corresponds to a world. Queries are formulas which are interpreted at worlds. A document is relevant to the query if $M \models \Diamond q[d]$. This means that a document d is relevant to a query q if there is some world connected to d in which q is true. Intuitively, worlds connected to d might represent other documents which are similar to d. In this way, the relevance test is not rigid because it evaluates the query not only in the world d but in other *similar* worlds.

Crestani and van Rijsbergen [14] proposed to apply logical imaging to IR for estimating the probability of relevance of a document with respect to a query. Imaging is a process developed in the framework of modal logic. It enables the evaluation of a conditional sentence without explicitly defining the operator $\rightarrow$. According to the possible-world semantics the truth value of the conditional $x \rightarrow y$ in a world w is equivalent to the truth value of the consequent y in the closest world w_x in which the antecedent x is true. From this process Crestani and van Rijsbergen implemented the logical uncertainty principle as follows. They used an extension of imaging that introduces a probability distribution on the set of worlds. The probability of a world can be regarded as a measure of the prior uncertainty associated with the beliefs in the world. In order to evaluate a conditional $x \rightarrow y$ there is a shift of the original probability P of the world w to the closest world w_x where x is true, i.e. the probability is moved from *not-x-worlds* to *x-worlds*, creating a new probability distribution P'. This process is called *deriving* P' *from* P *by imaging on* x. The set of index terms is considered as the set of the possible worlds. A document d is true in a world (term) if the term occurs in d. In an analogous way, a query q is true in a world (term) if the term occurs in q. A measure of similarity between terms is used to determine the accessibility relation between worlds. In order to evaluate the implication $d \rightarrow q$, a process of imaging on d is fired. This means that the probabilities from terms not occurring in the document d are transferred to terms occurring in the document d. The model of retrieval by logical imaging was later generalized to consider general imaging [15]. This extension aimed at overcoming the assumption related to the uniqueness of the world w_x, that is the uniqueness

of the world most similar to w where x is true.

Models based on preference logic [6] view preferential structures as being a natural choice for underpinning IR. A ranking of possible worlds, partially ordered by the preferences inherent in the given information need is defined. The minimal elements of the ordering correspond to the documents the user most prefers. The goal of the IR mechanim is to discover and deliver the ranking that corresponds to the user's information need. Relevance feedback techniques can be applied to acquire user preferences for information. The formalization of these preferences can be modeled through defaults and preclusion relationships.

Situation theory is a theory of information which provides an analysis of the concept of information and the manner in which intelligent organisms handle and respond to the information picked up from their environment. The nature of information flow and the mechanisms that give rise to such a flow are defined. The flow of information is the phenomenon where an object contains information about another object. Situation theory has been used to develop IR models. If the objects are documents and queries, then the exploitation of the flow of information between these objects is the task of an IR system. Specifically, determining relevance consists of computing the information contained in one object (a document) about another object (a query), i.e. the flow of information between the two objects. This was the approach followed in many works such as [35]. An IR model based on situation theory can be very expressive because it captures many aspects of information and it can cope with partiality and uncertainty.

Meghini and other researchers [45] proposed the use of terminological logics to model IR. Terminological logics are knowledge representation languages in which the tradeoff between expressiveness and tractability has been largely studied. The inherent expressiveness of these logics allows to construct structured representations for documents and queries. Subsumption is the basic inference mechanism in terminological logics. The approach proposed by Meghini and his colleagues models retrieval as subsumption: a document is considered relevant to a query if the representation of the query is subsumed by the representation of the document.

1.3.2 Discussion

Many works have focused on the use of logic to model retrieval problems. The main aim is to create expressive and uniform IR models able to improve retrieval effectiveness. In this sense, logic was found as a promising tool to capture the semantics of documents and information requests in a better way. In last section we presented a number of approaches addressing these issues. Many other approaches exist in the literature. The variation in approaches reflects different vehicles deemed suitable for modeling IR. Nevertheless, no consensus has been reached regarding what the best technique is [34].

Although substantial theoretical progress has been made, the use of logic for IR modeling is still in its early stages [34]. Further investigation and development are required before the effectiveness of logical models can be established. Besides, more experimental work is necessary to demonstrate the effectiveness of logical models of IR. Usually, the implementation of logical models is complex and many logical models have not even tried to propose implementations for their tasks. However some experimentations on standard test collections have been accomplished. For instance, the model of IR based on logical imaging has been tested on large document bases [13]. Since the model based on logical imaging is more general than classical models, it introduces additional technical problems when evaluating. As a result, some simplifications and reductions should be adopted. This is not a particular drawback of this model but it is a general consideration for logical models.

1.4 Our view

Information Retrieval is a very classical research field in which theory and practice are mixed up. Some approaches to retrieval system design are strongly guided by theory. Other have little real theorical underpinning but are instead more experimental and *ad-hoc* in character. For many years, there have been controversy between the people who is mainly inspired by theory and those who are influenced more by experimentation. Theory-guidedness is a good thing if the theory leads to promising retrieval rules to try out. Indeed, good theories can provide inferential power which can help to minimize empirical floundering. Nevertheless, having to stay within the constraints of a strict theoretical formalism can also impose costs and penalties.

In recent years, evaluation in IR has increasingly received more weight. Evaluation itself is a dominant topic in IR and some conferences, like the TREC conference, focus solely on this subject. Even in general conferences like the ACM SIGIR, IR evaluation arises as a fundamental issue. Evaluation procedures are based on comparisons using statistical values such as recall and precision. Although the utility of statistical values is certainly unquestionable, theoreticians claim that there is a need for a more formal characterization of IR systems. To be able to make more strict statements concerning the quality of an IR model, we have go further and apply more formal means of comparison. Furthermore, to prove specific statements concerning the behavior of IR systems, statistical tests are not adequate.

We have chosen a theoretical approach. We strongly believe that retrieval models can benefit from the formal grounds of a good theory. However we do not pretend to disregard the importance of evaluation methods. We are aware that retrieval models should be evaluated against standard benchmarks in order to clarify whether or not they are applicable within realistic systems and, in the case, foresee their retrieval performance. In this sense, we have taken great care of the complexity of the tasks involved in the model presented here. We have studied in depth the procedures that implement our proposals and, as a result, we propose efficient algorithms for the associated tasks. These algorithms have allowed us to build a prototype logic-based IR system that was used to carry out a proper evaluation against standard IR test collections. Sometimes, logical approaches are criticized because they are considered too abstract and far from the IR reality. In this respect, the methodology we have adopted for our research is twofold. First, we have modeled some classical problems within our framework. This objective aims at showing that the logical model can cope with classical tasks. Second, we emphasize the advantages of the use of our model. For instance, as well as handling classical representations as vectors, the model can cope with more expressive representations. In this respect, the model is general because it subsumes classical representations but it goes further. Besides, as it will be shown in the evaluation section, the increased expressiveness leads to an increment in the retrieval performance of the system. Since the framework is simple enough, we are able to extract automatically logical representations for documents and queries. Depending on the way we build the final logical formulas we can get different degrees of expressiveness and, hence, we can compare the retrieval performance of the system accordingly.

We will represent documents and queries as propositional formulas. These formulas provide us with a method to articulate expressive representations for documents and queries. For instance, a document can be represented as $(neural \wedge networks) \vee (genetic \wedge algorithms)$, where the topics of the document are clearly separated. Areas like Image Retrieval or Speech Retrieval need representations more expressive than classical vectors. Image recognition systems often supply results with a degree of vagueness. In these situations a more expressive formalism, in which several alternatives can be articulated, is desirable. A propositional formula can also be used

to represent the output of a speech recognition system. The same piece of speech can produce different sentences depending on the quality of the recognition system and the similarity between the pronunciation of the sentences. In these cases, a propositional representation is more precise than a vector. The use of negations gives us the ability to write accurate representations. An image can be represented by the formula $(deer \wedge \neg horse) \vee (\neg deer \wedge horse)$ and, hence, the semantics of the image is clearly specified. Information needs can also take advantage from the structure of propositional formulas. Indeed, negations in queries are a potential precision-oriented mechanism.

Expressive representations are usually needed by classical works. For instance, several works in Passage Retrieval [27, 9] have shown that a system can get better performance if it represents documents divided into several parts. Since the expressiveness of classical IR systems is limited, this sort of extensions have often been done using *ad-hoc* techniques. On the contrary, logic allows us to include split representations in an homogeneous way.

Logical models are more general. Last paragraphs have illustrated how logical expressions can represent classical documents and queries and, besides, how more expressive representations can be handled. This is an example of the generality that can be achieved when applying logic to model IR problems. Furthermore, the framework provided by logic allows us to encompass distinct features of the target domain in an homogeneous manner. In this respect, as well as representing documents and queries we can use propositional logic to model other notions such as thesaurus information, relevance feedback information and so forth. Since the same framework is used we can benefit from the inference capabilities of logic in order to reason about the domain. On the other hand, classical models with *ad-hoc* extensions often include distinct notions using distinct frameworks. This heterogeneity leads to complicated models in which it is hard to reason. Moreover, to carry out further extensions on heterogeneous models is very difficult.

In this thesis we have modeled situational information and relevance feedback information using propositional logic. Hence, the model is general because it is able to represent documents, queries, situational information and feedback information. This is important because we can apply the same techniques for distinct elements. For instance, the same procedures that are used to get a measure of relevance between documents and queries can be applied when situational information is introduced. The use of logic gives us the ability to generalize the model in an homogeneous way. Besides, the inherent expressiveness of logical expressions is a good tool for modeling relevance feedback information and situational knowledge. In relevance feedback it is usual to disregard the importance of documents which are marked by the user as unrelevant. Nevertheless, this kind of information can be very useful as a precision-oriented mechanism. Classical models do not have the appropriate tools to deal with negative feedback information. On the contrary, logic allows us to include negative information through negations. The use of negations leads to improvements in retrieval performance, as it will be shown in chapter 5.

We have presented a number of argumentations in favor of the use of logic to model IR problems. Logic is a powerful mechanism to formalize knowledge. If there is ever to be a major breakthrough in IR, it will come from approaches which model knowledge in a better way. We strongly believe that logic is a promising tool for attempting to achieve this goal. We have bridged classical IR by modeling classical problems within a logical framework. In a second step, we implemented efficiently the tasks involved in the model. These two objectives aim at motivating the classical IR community in favor of logical models. Then, we show the advantages of using logic. We believe that the model presented here can stand on the basis of future extensions which capture well the knowledge implicit in documents and queries. Although this is a great challenge we think that logic has the appropriate mechanisms to start the quest.

1.5 Outline of this thesis

The first part of this thesis (chapter 2) is dedicated to present the basic model that we propose. Chapter 3 deals with the implementation and evaluation of the model. The first part of this chapter is focused on the design of efficient procedures to implement the computation of similarity between documents and queries. Section 3.3 presents the evaluation of the model against standard IR collections. Subsection 3.3.1 gives some background on IR evaluation. Those who are familiar to IR evaluation can skip this subsection. Nevertheless, we recommend to read the subsection *Test collections* of section 3.3.1, where we motivate our decision to use four small collections. Chapter 4 deals with the introduction of retrieval situations within the model. Readers acquainted with conditional logics might find familiar concepts but all the sections are needed to understand our proposal. In chapter 5 we present the model of feedback. The first part of this chapter (until the beginning of section 5.1.1) gives a brief introduction to feedback. Those who are familiar to feedback can go directly to section 5.1.1. Chapter 6 presents the extension of the model to deal with term similarity and inverse document frequency. This chapter can be read before reading chapters 4 and 5 because it is an extension of the model presented in chapter 2 and evaluated in chapter 3.3. The thesis ends with some conclusions.

There are a number of publications derived from the work presented here. The basic model (chapter 2) was presented as a full research paper [39] at *SIGIR-99, the 22nd ACM Conference on Research and Development in Information Retrieval.* The implementation of the model was presented in [41, 40]. Specifically, in *SPIRE-2000, the 7th Symposium on String Processing and Information Retrieval* we presented the algorithms BRsim-1C and BRsim-SC [41] (sections 3.1.2 and 3.1.1). Algorithm A_BRsim-SC, which computes similarity in an efficient way (section 3.1.5), was presented at the *SIGIR-2000 Workshop on Mathematical and Formal Methods in Information Retrieval* [40]. The introduction of retrieval situations within our model [42] was published in *LUMIS'2000, the DEXA'2000 International Workshop on Logical and Uncertainty Models for Information Systems.* The model of feedback and its evaluation results (chapter 5) were presented in the poster session [43] of *SIGIR-2001, the 24th ACM Conference on Research and Development in Information Retrieval.*

Chapter 2

Representation and matching

Logical models of Information Retrieval (IR) have not been very successful. These models are often complex and, as a consequence, their actual applicability is unclear. However, generality and expressiveness, which are intrinsic properties of logic, are desired characteristics for an IR system. In fact, there is a demand for the development of new formalisms able to model IR systems in a more generic manner [34]. Classical models tend to consider simplistic standpoints. Particularly, representations of documents and queries are often built under too strict assumptions. For instance, it is usual to take a total knowledge assumption: it is assumed that we do actually know whether or not a document deals with every index term. Clearly this is an unrealistic standpoint because it is not feasible to extract the topics of a document from word counts and statistical techniques. The indexing process is inherently vague and a formalism allowing uncertainty can be very helpful.

Current practice in IR establishes a separation between representation and reasoning [57]. As a matter of fact, the same method for computing the representations of documents and queries (e.g. tf/idf weighting scheme) is being used with widely different matching techniques. On the contrary, logic provides us with a framework in which representation and reasoning are not independently motivated [57].

In this chapter we show that some IR classical tasks can be accomplished within a logical framework. Since it is often assumed that logic is far from IR reality, the first objective tries to convince the reader that classical problems can be modeled by a logical model.

We use Propositional Logic for representing documents and queries. This formalism is expressive enough to represent binary-weighted classical vectors. Besides, propositional languages can cope with more expressive representations. We found that Belief Revision (BR) is very useful for IR. BR captures formally the notion of proximity between logical formulas. In this way we can compute a measure of similarity between documents and queries represented as logical formulas. This gives us a measure of how much $d \models q$, where d and q are the logical representations of a document and a query respectively and $\models$ represents the classical entailment. In an analogous way, a measure of how much $q \models d$ can be defined. This measures captures the notion of *specificity*, i.e. how specific is the document with respect to the query.

2.1 Basic Model

Our approach is based on modeling documents and queries as propositional formulas. Although the expressiveness of Propositional Logic is somewhat reduced, we think Propositional Logic is

a good starting point because it allows to address particular cases of classical IR problems and its limited expressiveness benefits the computational complexity of the tasks involved. Besides, Propositional Logic is enough to reveal some of the advantages of a logical approach and, taking into account the tradeoff between expressiveness and complexity, the applicability of the results to more expressive logics could be studied. Furthermore, we want to evaluate our model through experimentation and, hence, we need a method that constructs automatically the representations of documents and queries. The state of the art of Natural Language Processing techniques for IR has not evolved enough to be able to get complex expressions, such as First Order Logic formulas, from plain text. Thus, we will use classical IR techniques to extract significant terms from documents and queries and these terms will be represented within propositional formulas. The next section presents some preliminaries needed to understand the rest of the chapter.

2.1.1 Preliminaries

A propositional alphabet $\mathcal{P}$ is a set containing all the propositional letters in the propositional language. Formulas are built over $\mathcal{P}$ using the connectives $\neg$ (not), $\vee$ (or), $\wedge$ (and). Additional connectives are used as shorthands, $\alpha \supset \beta$ denotes $\neg\alpha \vee \beta$ and $\alpha \equiv \beta$ is a shorthand for $(\alpha \wedge \beta) \vee (\neg\alpha \wedge \neg\beta)$. Semantics is given by *interpretations*. An interpretation is a function from the propositional alphabet to the set {true,false}. Given an interpretation I (the valuation for each propositional letter) we can get the valuation for any propositional formula as follows. For the sake of simplicity we denote the valuation for formulas with the same name than the valuation for letters.

- $I(\neg\alpha) =$ true if $I(\alpha) =$ false and $I(\neg\alpha) =$ false if $I(\alpha) =$ true.
- $I(\alpha \wedge \beta) =$ true if both $I(\alpha) =$ true and $I(\beta) =$ true hold; otherwise $I(\alpha \wedge \beta) =$ false.
- $I(\alpha \vee \beta) =$ false if both $I(\alpha) =$ false and $I(\beta) =$ false hold; otherwise $I(\alpha \vee \beta) =$ true.

A *model* of a formula α is an interpretation mapping α into true. Throughout this thesis we denote interpretations by the set of letters which are mapped into true. For instance, given the alphabet $\mathcal{P}=\{a, b, c\}$, the model which maps a and c into true and b into false is denoted by $\{a, c\}$. A *theory* is a set of formulas. An interpretation is a model of a theory if it is a model of every formula of the theory. Given T a propositional formula or a theory, the set its models is denoted by $Mod(T)$. Given two formulas α and β we say that α entails β, written $\alpha \models \beta$, if β is true in every model of α. An *atom* is a propositional letter and a *literal* is a propositional letter or its negation. Let A and B be two sets, the symmetric difference between them, $A \triangle B$, is given by $(A \cup B) \setminus (A \cap B)$, where $\setminus$ is the regular difference between sets. A *preorder* over a set $\mathcal{I}$ is a reflexive and transitive relation on $\mathcal{I}$. A preorder $\leq$ is *total* if for each $I, J \in \mathcal{I}$ either $I \leq J$ or $J \leq I$ holds.

2.1.2 Representation of documents and queries

In our model the indexing vocabulary of an IR system is represented by a propositional alphabet. Each propositional letter represents an index term. Documents and queries are represented by propositional formulas. For instance, given a propositional alphabet with the form $\mathcal{P} = \{algebra, calculus, ...\}$, a document represented by the formula *algebra* is a document dealing with algebra and a document represented by $\neg calculus$ is a document which does not deal with calculus. A

Classical Representation	Logical Representation
Indexing vocabulary: $\{t_1, t_2, t_3, t_4\}$ Document: (0,1,0,1) Query: (1,0,0,1)	Propositional Alphabet: $\{t_1, t_2, t_3, t_4\}$ $\neg t_1 \wedge t_2 \wedge \neg t_3 \wedge t_4$ $t_1 \wedge t_4$

Figure 2.1: An example of the translation from binary-weighted vectors to logical formulas

query represented by $algebra \wedge \neg calculus$ is asking for documents dealing with algebra and not dealing with calculus.

The expressiveness of Propositional Logic allows us to articulate logical representations for classical vectors with binary weights. A binary-weighted vector $\vec{v} = (w_1, w_2, \ldots, w_n)$ can be represented as a logical formula on a propositional alphabet $\mathcal{P} = \{t_1, t_2, \ldots, t_n\}$, where each propositional letter represents an index term of the vector space. Specifically, vectors $\vec{v}$ representing documents are translated into logic as conjunctive formulas with the form $l_1 \wedge l_2 \wedge \cdots \wedge l_n$, where $l_i = t_i$ iff $w_i = 1$ and $l_i = \neg t_i$ iff $w_i = 0$. Query vectors receive a different treatment. The vector space model does not allow to articulate negations. This means that a 0-weight for a term t in a query vector represents the fact that the user is not interested in the term t. If we represent $\neg t$ in the logical formulation we would be asking for documents which do not deal with t. On the contrary, the intended behavior is that t is not important for characterizing the user's interests. Then, a query vector $\vec{q} = (w_{q1}, w_{q2}, \ldots, w_{qn})$ should be represented in logic as follows. Let $i, j, \ldots$ be the k indexes of the query vector having weight equal to 1, the formulation of the query within Propositional Logic is $t_i \wedge t_j \wedge \ldots$. The terms with weight 0 within the query vector do not appear in the logical representation. The logical framework is intrinsically more expressive because we can articulate queries like $maths \wedge \neg algebra$. These queries cannot be articulated in the vector space model. Figure 2.1 shows an example of the translation into logic of a document and a query vector.

As well as being able to model classical vectors, Propositional Logic gives us the ability to articulate representations which are not usual in classical systems. We can express *partial* vectors to represent documents. This means that we do not have to store information about all the index terms when representing a document. For instance, we can store representations like $\neg t_1 \wedge t_3 \wedge \ldots \wedge t_n$, where some alphabet letters are missing (t_2). This models the lack of information with respect to the missing term: we do not know whether or not the document actually deals with the concept associated to the missing term.

The use of disjunctions in the representations of documents allows us to express several views of a document. For instance, we can distinguish between different parts of a document: title, abstract, etc. This can be done using propositional formulas like $(\neg t_i \wedge t_j \wedge \ldots \wedge t_k) \vee (\neg t_l \wedge t_m \wedge \ldots \wedge \neg t_h)$, where each conjunction represents a different view of the document. The idea of using split document representations can be very helpful in IR. As a matter of fact, in the area of Passage Retrieval several works [27, 9] have isolated parts of the documents that are supposed to be semantically different. At a first attempt one can divide a full-text document into its paragraphs. More complex techniques use windows of text of different sizes and statistical computations in order to get a document split into chunks. The evaluation results of these methods show improvements when the structure of document's representations contains several

parts. In this sense, Propositional Logic is a general formalism because it allows us to represent a whole vector but it also can cope with document's representations divided into parts.

In Image Retrieval and Speech Retrieval propositional formulas can also be very helpful. A process of analysis of an image can suggest that a shape is a deer or a horse. In this case a vector representation impose a simplification. On the contrary a logical representation gives us the ability to write expressions like $(deer \wedge \neg horse) \vee (\neg deer \wedge horse)$. In the same line, after scanning a piece of speech, a voice recognition system can be unable to distinguish between "How to wreck a nice beach" or "How to wreck an ice peach" or "How to recognize speech". Again, a split representation seems to be useful to represent several alternatives.

Negations can also play an important role in actual systems. Negations in queries act as a valuable precision mechanism. Even in standard IR test collections, some queries lead naturally to negated terms. For instance, the fourth query of the CACM collection contains the sentence "We are not interested in the dynamics of arm motion". Clearly, negations should be used to express this information need. Classical systems use to reduce unfairly the role of negations. On the contrary, our model handles positive and negative terms in the same way. In Image Retrieval negations can help to establish precisely the information that we have about a picture. For instance, let us recall the previous example, in which an image is represented by the formula $(deer \wedge \neg horse) \vee (\neg deer \wedge horse)$. In this representation the appearance of negations is fundamental for the meaning of the document.

An important advantage is that the model is *homogeneous* because the query language is the same as the document language. In [10] it was shown that homogeneous models fulfill some interesting properties. For instance, we can define inverse relations between documents and queries within an homogeneous model. In this sense, if we have a measure of the uncertainty of $d \models q$ we can also define a measure of the uncertainty of $q \models d$. Later on, we will show the usefulness of a measure of the uncertainty of the entailment $q \models d$.

One of the main drawbacks of the formalism is that non-binary weights cannot be handled. However there are several domains where binary representations make sense. For example, a lot of features of images are binary and a binary representation seems also to be appropriate when representing metadata information. Furthermore, in chapter 6 we will present an extension of the model to handle term similarity and inverse document frequency.

2.1.3 Matching

In this section we define a measure of relevance between documents and queries represented as propositional formulas. A direct approach would apply the notion of logical entailment: given d and q the propositional formulas representing a document and a query respectively, the document is relevant to the query if and only if $d \models q$. However this criterion is too strict because it does not consider partial matching. In fact, we can get a direct translation of the Boolean Model (BM) using Propositional Logic for representing documents and queries and using the logical entailment for deciding relevance. Boolean queries are translated directly into propositional logic formulas and the translation of the BM document's representation (bags of terms) into logic is completely analogous to the translation of binary-weighted vectors, which was sketched above. Anyway, a model considering partial relevance is preferable. In this line, the Logical Uncertainty Principle (LUP) defined by van Rijsbergen [61] establishes the basis for any logical approach that tries to capture the notion of partial relevance:

> Given any two sentences x and y; a measure of the uncertainty of $y \rightarrow x$ relative to a given data set is determined by the minimal extent to which we have to add

information to the data set, to establish the truth of $y \rightarrow x$.

Since this foundational work, several researchers have utilized different formalisms in order to implement this principle. The studies [12] and [34] are interesting compendia of the work developed in the area. Our approach applies techniques of the field of Belief Revision in order to get a non-binary measure of the entailment $d \models q$. Our *measure of the minimal extent* (in the sense of the LUP) is obtained from computations of distances between logical models of d and q. Given an existing logical knowledge base and a new information, Belief Revision copes with the problem of accommodating the new information into the knowledge base. If contradiction arises, the old knowledge has to be changed *minimally* in order to accept the new information in a consistent way. Distinct BR techniques use distinct methods which define a measure of proximity between old and new knowledge. We find that the idea of proximity present in Belief Revision is appropriate for our purposes because we can apply BR techniques and obtain a measure of proximity between a document and a query expressed as logical formulas. The next section presents the basic foundations of Belief Revision.

Belief Revision

Belief Revision handles the problem of updating knowledge bases after the acquisition of new information. The most interesting case arises when the acquired information is in conflict with the current knowledge base. It is well known that inconsistency is a grave problem because one can deduce *anything* from a set of inconsistent premises. Since the updated knowledge base should be consistent, the revision process involves usually a method to delete some old information. In order to decide what information is prepared to give up, Belief Revision is able to model rational decisions concerning modifications to a knowledge base. A rational reasoner is expected to modify its knowledge base as little as possible in order to accommodate new information. This intuition is formalized by the *Principle of Minimal Change*, which states that as much old beliefs as possible should be conserved.

Along this thesis we will consider knowledge bases represented as *belief bases*, i.e. sets of logical sentences which are not closed under logical consequence. Belief Revision methods are sometimes characterized using *belief sets* to represent knowledge bases. A belief set is a set of sentences which is closed under logical consequence. We have chosen the former representational approach because belief bases are easier to handle since they are finite structures.

BR methods can be divided into *formula-based approaches* and *model-based approaches*. Given a belief base K and a new information A, formula-based approaches select formulas syntactically appearing in the knowledge base K and in the new information A. The selected formulas form the revised belief base, which is denoted by $K \circ A$. On the other hand, model-based approaches operate on the logical interpretations. These methods use some form of *minimal changes* on the models of the belief base. An ordering on the set of all interpretations is defined and used to decide which interpretations should constitute the models of $K \circ A$. The intended meaning of such an ordering is that some interpretations that are models of A (but not of K) are closer to models of K than other interpretations. Such an ordering of interpretations should, of course, be dependent on the belief base K. In our work we focus on a model-based approach to BR and on propositional languages.

A BR operator, denoted by $\circ$, takes a theory K and a new formula A and builds the updated theory $K \circ A$. We propose to apply Dalal's revision operator [17], which is a model-based operator, because there are interesting connections between the semantics of this BR operator

and the behavior of classical IR matching functions. Indeed, we can capture the behavior of classical IR matching functions within a BR process.

Model-based approaches to BR select models of the revising formula A on the basis of some notion of proximity to the models of the theory K. The selected models make up the set of models of the revised theory. Specifically, Dalal's revision operator, denoted by $\circ_D$, uses the number of propositional letters on which two interpretations I and J differ as a measure of *distance* between them, $dist(I, J)$. As we represent interpretations by the set of letters mapped into true, a measure of distance between two interpretations can be directly obtained from the cardinality of the symmetric difference between their respective sets. A measure of distance between the set of models of a formula ψ and a given interpretation I is defined as the distance from I to its closest interpretation in $Mod(\psi)$:

$$Dist(I, Mod(\psi)) = min_{M \in Mod(\psi)} dist(M, I) \tag{2.1}$$

Let $\mathcal{I}$ be the set of all propositional interpretations on the propositional alphabet. A preorder $\leq$ over $\mathcal{I}$ is a reflexive and transitive relation on $\mathcal{I}$. We define $<$ as $I < J$ if and only if $I \leq J$ and $J \not\leq I$. A preorder is *total* if for every $I, J \in \mathcal{I}$, either $I \leq J$ or $J \leq I$.

Given a formula ψ, a total preorder between interpretations can be extracted from the nearness of each interpretation to the set of models of the formula ψ. Dalal defines a total preorder $\leq_\psi$ as:

$$I \leq_\psi J \quad \text{iff} \quad Dist(I, Mod(\psi)) \leq Dist(J, Mod(\psi)) \tag{2.2}$$

Intuitively, $I \leq_\psi J$ means that I is closer to $Mod(\psi)$ than J.

When revising a theory ψ with a new information μ, the models of the new information which are the closest to the theory (w.r.t the order induced by the theory) are selected to be the models of the revised theory:

$$Mod(\psi \circ_D \mu) = Min(Mod(\mu), \leq_\psi) \tag{2.3}$$

There is a number of circumstances favoring the use of Dalal's operator. Katsuno and Mendelzon [31] analyzed a number of revision operators in the framework of the AGM postulates and concluded that Dalal's operator is the only one that fulfills the original AGM postulates in a non-trivial way. The revision operators proposed by Fagin, Ullman and Vardi [21], Borgida [5], Winslett [65], Satoh [55] and Weber [62] fail to satisfy the postulates. The AGM postulates are a set of rationality postulates that the operations of revision must satisfy. These postulates were originally defined by Alchourrón, Gärdenfors and Makinson in [2] and have been widely recognized as the basis of any revision method. If a revision operator satisfies the AGM postulates then it is assured that the operator produces a revision with minimal change. This is an important factor for our application within IR. Next paragraphs explain formally why Dalal's operator fulfills the AGM postulates.

Consider a function that assigns to each propositional formula ψ a preorder $\leq_\psi$ over the set propositional interpretations. For instance, in equation 2.2 an assignment was defined according to the distance to the set of models of the formula ψ. Katsuno and Mendelzon defined the property of faithfulness for assignments. This property is fulfilled if the following three conditions hold:

- If $I, I' \in Mod(\psi)$, then $I <_\psi I'$ does not hold.
- If $I \in Mod(\psi)$ and $I' \notin Mod(\psi)$, then $I <_\psi I'$ holds.
- If $\psi \equiv \phi$, then $\leq_\psi = \leq_\phi$.

That is, a model of ψ cannot be strictly less than any other model of ψ and must be strictly less than any non-model of ψ. Following this definition Katsuno and Mendelzon proved the next theorem.

> **Theorem:** Revision operator $\circ$ satisfies the postulates (R1)-(R6) if and only if there exists a faithful assignment that maps each knowledge base ψ to a total preorder $\leq_\psi$ such that $Mod(\psi \circ \mu) = Min(Mod(\mu), \leq_\psi)$.

The postulates (R1)-(R6) are a set of postulates defined by Katsuno and Mendelzon which are equivalent to the original AGM postulates. AGM postulates do not assume any concrete representation of the knowledge base whereas Katsuno and Mendelzon's postulates assume knowledge bases represented by a finite set of propositional sentences.

It can be easily proved that the assignment used by Dalal's operator (equation 2.2) is faithful. Next lines sketch this proof.

- Give I and I' two models of ψ both $I \leq_\psi I'$ and $I' \leq_\psi I$ hold. This is because $Dist(I, Mod(\psi)) = Dist(I', Mod(\psi)) = 0$. Then, $I <_\psi I'$ does not hold.
- If $I \in Mod(\psi)$ and $I' \notin Mod(\psi)$, then $Dist(I, Mod(\psi)) = 0$ and $Dist(I', Mod(\psi)) > 0$. Hence, $I <_\psi I'$ holds.
- If $\psi \equiv \phi$, then $Mod(\psi) = Mod(\phi)$ and, thus, $\leq_\psi = \leq_\phi$.

Since Dalal's assignment is faithful, we can follow the previous theorem and conclude that Dalal's revision satisfies the BR rationality postulates.

There are three reasons why methods from the literature fail to satisfy these postulates. First, some approaches use partial orders instead of total orders. In some cases, it might be reasonable to suppose that certain interpretations will be incomparable in terms of their closeness to the knowledge base. For our application, a total order is more appropriate because we can thus compare any two interpretations in terms of their distance to the query. Second, the use of orders that depend both on the knowledge base and the revising formula. At this point, we think that the order should depend only on the query. An order which depends both on query and document could be applied for other IR tasks. Nevertheless, this kind of orders are more difficult to justify intuitively and lead to somewhat unnatural orderings [31]. Third, some methods use pointwise orderings that are induced by each interpretation of the knowledge base, instead of by the knowledge base as a whole. This sort of methods are semantically different to revision methods and they are not suitable for modeling the matching process between a document and a query.

Other key consideration in favor of Dalal's operator is that there are very interesting results with regard to its complexity. In particular, the work by Eiter and Gottlob [20, 19] analyze the computational properties of different revision approaches. They focus on the following implication problem: given a knowledge base T, a revising formula p and a formula q, decide whether q is derivable from $T \circ p$, the updated knowledge base. The behavior of different methods is compared

in terms of the complexity when facing this implication problem. Dalal's method can decide the previous problem in polynomial time as long as the involved formulas are Horn formulas and the size of the revising formula is bounded by a constant. A Horn formula is a formula which can always be written as a disjunction of literals with at most one positive literal.

The complexity of the model becomes specially important when dealing with IR problems. We have taken great care of this issue and in chapter 3 we present efficient algorithms for computing similarity between documents and queries.

The revision $q \circ_D d$

As well as the good properties of Dalal's operator for generic application, we found an interesting connection between Dalal's semantics and IR matching functions. Dalal's operator measures the distance between interpretations in terms of the number of differing propositional letters. Since the propositional alphabet models the indexing vocabulary, the distance between two interpretations is given by the number of differing indexing terms between the interpretations. Let us consider a query and a document represented as propositional formulas q and d respectively, and the revision process $q \circ_D d$. We can think about this revision process in the line of van Rijsbergen's uncertainty principle: $q \circ_D d$ represents the theory which accepts the content of the document changing the query as little as possible. Given this start point we should define a measure of that change in order to get a measure of the uncertainty of $d \models q$. Within the revision process $q \circ_D d$ a distance from any interpretation I to the set of models of the query is defined as follows:

$$Dist(I, Mod(q)) = min_{M \in Mod(q)} dist(M, I) \tag{2.4}$$

In an ideal situation, we would know exactly which is the interpretation that corresponds to the *actual world*, i.e. we would know exactly the set of topics that a document deals with and, thus, there would be an only interpretation satisfying d (i.e. d would have an only model). In classical systems it is usual that the document representation has information about presence or absence for all the index terms of the indexing vocabulary. On the logical side this corresponds with a document representation with a single model, which is the one mapping the terms appearing within the document into true and all the other terms into false. Of course, this is a simplification because a document does not deal with all the topics whose associated term(s) appear within the document. Nevertheless, this is widely accepted in IR and few models have gone further. If the document has a single model we can directly define a measure of distance from the document to a query using the distance from the single model of the document to the set of models of the query, which is defined within the revision process $q \circ_D d$:

$$distance(d, q) = Dist(m_d, Mod(q)), \tag{2.5}$$

where m_d is the single model of the document. From the IR perspective this distance measures the number of index terms that should be changed in the document in order to satisfy the query. Then, the distance can be regarded as a measure of the minimal extent in the sense of van Rijsbergen's LUP. Note that if the document satisfies the query (i.e. if $d \models q$) then $distance(d, q) = 0$. By definition of $d \models q$: $Mod(d) \subseteq Mod(q)$. Then, the single model of the document, m_d, is a model of the query ($m_d \in Mod(q)$). As a result $Dist(m_d, Mod(q)) = 0$ and $distance(d, q) = 0$.

Given a propositional alphabet $\mathcal{P}$ and the propositional formulas d_1, d_2 and q representing two documents and a query, fig. 2.2 depicts the process of computation of the measures of distance between the query q and the documents d_1 and d_2. The distances are obtained from distances between interpretations computed within the revision processes $q \circ_D d_1$ and $q \circ_D d_2$. First the symmetric differences between document and query models are computed. These sets contain the differing letters between the model of the document and query models. The individual model-to-model distances are the cardinalities of the symmetric differences and, finally, the distance from the model of the document to the set of models of the query is the distance from the document model to its closest query model. Note that the final values of distance reflect our intuitions: we have to change one term in d_2 in order to satisfy the query whereas d_1 directly satisfies the query.

$\mathcal{P}= \{a, b, c\}$
$d_1 = a \wedge \neg b \wedge c$
$d_2 = a \wedge b \wedge c$
$q = a \wedge \neg b$

Revision $q \circ_D d_1$

doc models → / query models↓	d_1 $\{a, c\}$
$\{a\}$	$\{c\}$
$\{a, c\}$	$\emptyset$

Revision $q \circ_D d_2$

doc models → / query models↓	d_2 $\{a, b, c\}$
$\{a\}$	$\{b, c\}$
$\{a, c\}$	$\{b\}$

(2.2.a) Symmetric differences between query and document models

Revision $q \circ_D d_1$

doc models → / query models↓	d_1 $\{a, c\}$
$\{a\}$	1
$\{a, c\}$	0

Revision $q \circ_D d_2$

doc models → / query models↓	d_2 $\{a, b, c\}$
$\{a\}$	2
$\{a, c\}$	1

(2.2.b) Cardinalities of symmetric differences

$Dist(m_d, Mod(q)) = min\{1, 0\}$ $\quad$ $Dist(m_d, Mod(q)) = min\{2, 1\}$
$distance(d_1, q) = 0$ $\quad$ $distance(d_2, q) = 1$

Figure 2.2: Computation of the distance between documents and queries

The framework can handle representations of documents having more than one logical model. In this case we propose to use the average of the distances from each model of the document to the set of models of the query as the distance from the document to the query:

$$distance(d, q) = \frac{\sum_{m \in Mod(d)} Dist(m, Mod(q))}{|Mod(d)|} \tag{2.6}$$

If there are several interpretations satisfying the document then the most sensible decision is to consider that all of them have the same chance of being the one that corresponds to the

actual world (i.e. the one that corresponds to the actual contents of the document). That is why we compute the average distance from models of the document to the set of models of the query. Other measures of distance from the document to the query have been taken into account. We could have defined the distance from a document to a query considering only the distance from one model of the document. For instance we could take the distance from the *worst* model of the document, i.e. the one whose distance to the query is the largest. However, this selection can lead to counterintuitive results as shown in the following example. Let us consider the alphabet $L = \{a, b, c\}$, the query $q = a \wedge b$, and the documents $d_1 = a \wedge c$ and $d_2 = a \wedge \neg b \wedge c$. The approach that selects the worst model of the document would produce the same value of distance for both documents with respect to the query q. This is against the expected and intuitive behavior. A measure that takes into account all the models of the document is fair. In absence of information, the most sensible choice is to consider all the potential possibilities about the actual contents of a document. In the example presented before, if we do not know whether the document d_1 actually deals with b we should take into account both the model that maps b into true and the model mapping b into false.

A total preorder $\leq_q$ induced by the query can be defined. This preorder establishes formally a method to rank documents in terms of their respective distances to the query.

$$d_i \leq_q d_j \text{ iff } distance(d_i, q) \leq distance(d_j, q) \tag{2.7}$$

We can also define $<_q$ as $d_i <_q d_j$ iff $d_i \leq_q d_j$ and not $d_j \leq_q d_i$, and $=_q$ as $d_i =_q d_j$ iff $d_i \leq_q d_j$ and $d_j \leq_q d_i$. Thus, within the revision process $q \circ_D d$, we obtain a formalization of a rank of documents given a query. For the example depicted in fig. 2.2, $d_1 <_q d_2$ holds.

The distance can be directly transformed into a similarity measure in the interval $[0, 1]$. The next equation defines the similarity measure $BRsim$, which is a normalization of *distance* which takes into account that the number of letters appearing in the query is an upper bound for *distance*:

$$BRsim(d, q) = 1 - distance(d, q)/k, \tag{2.8}$$

where k is the number of propositional letters appearing in q. Figure 2.3 shows an example of the computation of $BRsim$ for a document having several models.

An important circumstance is that this similarity measure provides us with a method to know whether or not $d \models q$ holds. The next theorem shows the correspondence between the test using the classical entailment and the measure of similarity $BRsim$.

$$\textbf{Theorem}: \; d \models q \text{ iff } BRsim(d, q) = 1. \tag{2.9}$$

$\mathcal{P}$= $\{a, b, c\}$
$d = a \wedge c$
$q = a \wedge b$

document models →	m_1	m_2
query models↓	$\{a, b, c\}$	$\{a, c\}$
$\{a, b\}$	$\{c\}$	$\{b, c\}$
$\{a, b, c\}$	$\emptyset$	$\{b\}$

(2.3.a)Symmetric differences between query and document models

document models →	m_1	m_2
query models ↓	$\{a, b, c\}$	$\{a, c\}$
$\{a, b\}$	1	2
$\{a, b, c\}$	0	1
$Dist(m_i, Mod(q))$	0	1

(2.3.b)Cardinalities of symmetric differences and distance

$$distance(d, q) = \frac{0+1}{2} = 0.5$$

$$BRsim(d, q) = 1 - 0.5/2 = 0.75$$

Figure 2.3: Computation of BRsim between a document and a query

Proof:

"$\Rightarrow$"

By definition of $d \models q$: $Mod(d) \subseteq Mod(q)$. The measure $distance(d, q)$ is defined as the fraction $\frac{\sum_{m \in Mod(d)} Dist(m, Mod(q))}{|Mod(d)|}$. Each $m \in Mod(d)$ belongs also to $Mod(q)$ and, thus, $\forall m \in Mod(d), Dist(m, Mod(q)) = 0$ and $distance(d, q) = 0$. Finally, $BRsim(d, q) = 1 - distance(d, q)/k = 1 - 0/k = 1$.

"$\Leftarrow$"

If $BRsim(d, q) = 1$, $distance(d, q)$ has to be equal to 0. Since $distance(d, q)$ is an average over values greater or equal to 0, it has to hold that $\forall m \in Mod(d), Dist(Mod(q), m) = 0$. This means that $\forall m \in Mod(d), m \in Mod(q)$. Since $Mod(d) \subseteq Mod(q)$, it holds that $d \models q$.

The similarity measure $BRsim$ outperforms significantly the test $d \models q$ when $BRsim(d, q) < 1$. In this case, $d \models q$ does not hold and the test based on the notion of logical consequence can just conclude that the document is non-relevant. On the other hand, $BRsim$ gives us a value in the interval $[0, 1]$ that is obtained in terms of the distances from the models of the document to the query. Clearly, there can be very different cases satisfying $d \not\models q$. Let us consider two situations: a) $d \not\models q$ because one model of the document is not a model of the query (consider that the distance from this model to the query is 1, i.e. its closest query model fares only one propositional letter to this model), all the other models of the document are models of the query. b) $d \not\models q$ because all the models of the document are not models of the query. Clearly, the document of the situation a) has more chance to be relevant to the user than the document of the situation b). Given a query q and two documents, d_1 and d_2, fig. 2.4 presents an example of this kind of situations. No one document satisfies the query, i.e. $d_1 \not\models q$ and $d_2 \not\models q$, but the measure $BRsim$ is able to distinguish between them. The second document has much more chance to be relevant to the user.

$\mathcal{P}$= $\{a, b, c\}$
$d_1 = \neg a \wedge \neg b \wedge c$
$d_2 = a \wedge c$
$q = a \wedge b$

$Mod(q) = \{\{a, b\}, \{a, b, c\}\}$

$Mod(d_1) = \{\{c\}\}$
$Mod(d_1) \not\subseteq Mod(q)$
$d_1 \not\models q$

$Mod(d_2) = \{\{a, c\}, \{a, b, c\}\}$
$Mod(d_2) \not\subseteq Mod(q)$
$d_2 \not\models q$

Revision $q \circ_D d_1$

doc models $\rightarrow$ query models$\downarrow$	d_1 $\{c\}$
$\{a, b\}$	$\{a, b, c\}$
$\{a, b, c\}$	$\{a, b\}$

Revision $q \circ_D d_2$

doc models $\rightarrow$	d_2	
query models$\downarrow$	$\{a, c\}$	$\{a, b, c\}$
$\{a, b\}$	$\{b, c\}$	$\{c\}$
$\{a, b, c\}$	$\{b\}$	$\emptyset$

(2.4.a) Symmetric differences between query and document models

Revision $q \circ_D d_1$

doc models $\rightarrow$ query models$\downarrow$	d_1 $\{c\}$
$\{a, b\}$	3
$\{a, b, c\}$	2
$Dist(m, Mod(q))$	2

Revision $q \circ_D d_2$

doc models $\rightarrow$	d_2	
query models$\downarrow$	$\{a, c\}$	$\{a, b, c\}$
$\{a, b\}$	2	1
$\{a, b, c\}$	1	0
$Dist(m, Mod(q))$	1	0

(2.4.b) Cardinalities of symmetric differences and distance

$distance(d_1, q) = 2$ $\quad$ $distance(d_2, q) = \frac{1+0}{2} = 0.5$

$BRsim(d_1, q) = 1 - 2/2 = 0$ $\quad$ $BRsim(d_2, q) = 1 - 0.5/2 = 0.75$

Figure 2.4: BRsim vs. logical entailment

If q and d are conjunctions of literals representing binary-weighted vectors $BRsim$ is equivalent to the inner product query-document matching function. Next paragraphs sketch the proof of this equivalence.

Let $\vec{d} = (w_{d1}, w_{d2}, \ldots, w_{dn})$ and $\vec{q} = (w_{q1}, w_{q2}, \ldots, w_{qn})$ be two binary-weighted vectors (i.e. $\forall i, w_{di}, w_{qi} \in \{0, 1\}$) representing a document and a query in the vector space model. The inner product query-document matching function measures the similarity between $\vec{d}$ and $\vec{q}$ as follows:

$$sim(\vec{d}, \vec{q}) = \frac{w_{d1}w_{q1} + \cdots + w_{dn}w_{qn}}{w_{q1} + \cdots + w_{qn}} \tag{2.10}$$

This classical matching process is directly modeled within our approach. Vectors representing documents and queries are translated into propositional logical formulas as presented in the previous section. Following that translation we demonstrate now that $BRsim(d, q) = sim(\vec{d}, \vec{q})$, where d and q are the translations into logic of the vectors $\vec{d}$ and $\vec{q}$ respectively.

Since vector weights are binary, the sum $w_{q1} + \cdots + w_{qn}$ is equal to k, the number of query terms appearing in q (recall that the formula q is built only with the terms having weight equal to 1 in $\vec{q}$). As a result we have to demonstrate that $k - distance(d, q)$ is equal to $w_{d1}w_{q1} + \cdots + w_{dn}w_{qn}$. The value of $distance(d, q)$ is defined as the fraction $\frac{\sum_{m \in Mod(d)} Dist(m, Mod(q))}{|Mod(d)|}$. The formula d is the translation of $\vec{d}$ into logic and, thus, it has a single model md, which maps the terms with weight 0 in $\vec{d}$ into false and the terms with weight 1 into true. Then, $distance(d, q) = Dist(md, Mod(q))$. Let us recall that $Dist(md, Mod(q)) = min_{mq \in Mod(q)} dist(mq, md)$. All the models of q map the letters appearing in the formula q (i.e. the terms with weight 1 in the vector $\vec{q}$) into true. Distinct models of q result from distinct combinations of the truth values assigned to the letters not appearing in q. Since the distance to the set $Mod(q)$ takes the minimum of the individual model-to-model distances, the letters not mentioned by q do not increase the distance. The letters appearing in q which are mapped by md into true do not increase the distance (because all the models of q map those letters into true as well). This means that the terms with weight 1 in both $\vec{q}$ and $\vec{d}$ do not increase the distance. Then, the final value of $Dist(md, Mod(q))$ is the number of letters appearing in q which are mapped by md into false (because all the models of q map those letters into true and md maps them into false). To sum up, $distance(d, q)$ is the number of terms with weight 1 in $\vec{q}$ and weight 0 in $\vec{d}$. Thus, $k - distance(d, q)$ is the number of terms with weight 1 in both vectors (k is the number of terms with weight 1 in $\vec{q}$). On the other hand, the sum $w_{d1}w_{q1} + \cdots + w_{dn}w_{qn}$ is also the number of terms having weight 1 in both $\vec{d}$ and $\vec{q}$.

The equivalence between $BRsim$ and the inner product query-document similarity measure is achieved when logical formulas are conjunctions of literals. This is a restricted case of our model, which is able to handle more expressive representations. For instance, if there is a query term not appearing in the document representation (this is not a regular assumption in classical systems) we do not assume that the document actually is (or is not) about that index term. This behavior is captured in the approach by the fact that there will be half of the models of the document with that query term positive and half of them with the term negative. From a quantitative point of view, these terms increase 0.5 the distance.

Extended boolean models (EBM) [54] preserve the query structure inherent in boolean models and incorporate weighted terms into both queries and documents. In this sense, one can see the extended boolean model with binary weights as a particular case of our model because: 1) binary-weighted vectors representing documents can be translated into propositional logic expressions, 2)

the model can also translate boolean queries into propositional logic expressions and 3) the model matches documents against queries using the inner product query-document similarity measure (which is a case of the EBM measure p-norm with p=1). Again, the model is more general because it can handle more expressive representations for documents. The p-norm measure is recognized as a well-behaved measure because it does not have the single operand dependency problem, the nonassociativity problem and the unequal importance problems. Lee [37] studied the mathematical properties of several operators proposed in the literature. These properties are very important because they affect retrieval effectiveness. Now we present some of the problems analyzed by Lee.

Let $d_1 = \neg information \wedge \neg retrieval$, $d_2 = information \wedge \neg retrieval$ and $q = information \wedge retrieval$ the representations of two documents and a query respectively. The boolean model would assign a similarity value of 0 for the two documents with respect to the query q. This result is only determined by the keyword *retrieval*, which is absent in both documents. Most people, however, will obviously decide than d_2 rather than d_1 is more similar to q. This example illustrates the single operand dependent property, which is fulfilled by some IR operators. Clearly, our similarity measure does not have this problem because all the keywords are taken into account to compute similarity.

No satisfaction of associativity implies that logically equivalent queries such as $q_1 = t_1 \wedge (t_2 \wedge t_3)$ and $q_2 = (t_1 \wedge t_2) \wedge t_3$ can produce different similarity values with respect to the same document. For our case, the associativity is directly obtained from the use of propositional logic. The previous two queries are logically equivalent and, thus, their sets of models are the same. This leads to the same values of the similarity measure $BRsim$ when matching against a given document.

An operator having the unequal importance problem violates the usual assumption that all the terms given in a query are equally important. Some fuzzy operators which consider only the terms with the highest and the lowest similarity to the query, have this problem. Again, the measure we have proposed gives the same importance to all the keywords in the query.

The revision $d \circ_D q$

The similarity measure $BRsim$ only takes into account how much of the query is satisfied by the document. It would be interesting to have a method to measure how much of the document is satisfied by the query. The distinction between these two notions was made explicit by Nie [47]. *Exhaustivity* is defined as the criterion measuring the number of query terms satisfied by the document, whereas *specificity* is the criterion that measures the number of document terms satisfied by the query. Exhaustivity is a recall-oriented criterion whereas specificity is a precision-oriented criterion. This means that exhaustivity gives more importance to retrieve relevant documents than to get a precise answer set (with few unrelevant documents). On the contrary, specificity takes care of having a precise answer set. The formal definitions of precision and recall will be presented in section 3.3.1. To motivate the usefulness of specificity let us consider the propositional alphabet $\mathcal{P} = \{algebra, calculus, trigonometry\}$, two documents and a query represented by the propositional formulas $d_1 = algebra$, $d_2 = algebra \wedge calculus \wedge trigonometry$ and $q = algebra$. A similarity measure which only considers the exhaustivity criterion would assign the same similarity value to both d_1 and d_2 with respect to q. It seems sensible to think that the first document is more relevant because it is more *specific* with respect to the query. The document represented by d_1 only deals with algebra, whereas the second document deals with some other issues. Therefore, it is reasonable to think that the first document has more

chance to be relevant to the user.

As a consequence, a good retrieval model should take into account both exhaustivity and specificity. The similarity measure $BRsim(d, q)$, which is computed within the revision process $q \circ_D d$, only considers the exhaustivity criterion. However, within the revision process $d \circ_D q$ we can get a measure of specificity. Note that this can be done because of the homogeneity of the model. Since documents and queries are represented using the same language we can interchange their roles and, hence, obtain a measure of specificity. A non-homogeneous model could not handle this reciprocal process.

The same technique that was used to get the similarity measure $BRsim(d, q)$ can be applied to the revision $d \circ_D q$ for obtaining a measure of specificity. In the revision $d \circ_D q$ we have a measure of distance $Dist$ from any interpretation I to the set of models of the document, which is defined as:

$$Dist(I, Mod(d)) = min_{M \in Mod(d)} dist(M, I) \tag{2.11}$$

Then, a measure of distance from the query to the document can be obtained as:

$$distance(q, d) = \frac{\sum_{m \in Mod(q)} Dist(m, Mod(d))}{|Mod(q)|} \tag{2.12}$$

Note that the previous formula is simply the reciprocal to the one we presented in the previous section. This distance counts the distance from a query to a document in terms of how specific is the document with respect to the query. Again, we can define an specificity-based measure of similarity, $BRsp$, in the interval $[0, 1]$ by normalization:

$$BRsp(d, q) = 1 - distance(q, d)/l, \tag{2.13}$$

where l is the number of propositional letters appearing in d.

In the previous example, it can be easily shown that $BRsim(d_1, q) = BRsim(d_2, q) = 1$. Figure 2.5 shows the process of computation of an specificity-based measure of similarity. The distances between interpretations used to get the final scores of similarity are computed within the revision processes $d_1 \circ_D q$ and $d_2 \circ_D q$. In the figure we use the letters a,c and t to refer to algebra, calculus and trigonometry respectively. Note that the final values of the measure of specificity reflect the intuition that d_1 is better than d_2.

In practice, we can compute the similarity between a document and a query taking into account exhaustivity and specificity. In [47] it was proposed a generic function combining both items. As we have a measure of exhaustivity ($BRsim$) and specificity ($BRsp$), we can define a similarity measure as a function of both with the form:

$$sim(d, q) = \alpha \times BRsim(d, q) + (1 - \alpha) \times BRsp(d, q), \tag{2.14}$$

where α is a tuning value in the interval $[0, 1]$ measuring the importance of exhaustivity vs. specificity. It seems clear that exhaustivity should be the dominant term in the final similarity value but the specificity criterion should not be left aside. Besides, there are other models where the integration of both criteria has been posed as a possible advantage [11]. If queries could be identified as recall or precision oriented, an adequate value of α could be set at retrieval time, leading to improvements in the overall performance of the system.

$\mathcal{P}= \{a, c, t\}$
$d_1 = a$
$d_2 = a \wedge c \wedge t$
$q = a$

Revision $d_1 \circ_D q$

query models → d_1 models↓	mq_1 $\{a\}$	mq_2 $\{a,c\}$	mq_3 $\{a,t\}$	mq_4 $\{a,c,t\}$
$\{a\}$	$\emptyset$	$\{c\}$	$\{t\}$	$\{c,t\}$
$\{a,c\}$	$\{c\}$	$\emptyset$	$\{c,t\}$	$\{t\}$
$\{a,t\}$	$\{t\}$	$\{c,t\}$	$\emptyset$	$\{c\}$
$\{a,c,t\}$	$\{c,t\}$	$\{t\}$	$\{c\}$	$\emptyset$

Revision $d_2 \circ_D q$

query models → d_2 models↓	mq_1 $\{a\}$	mq_2 $\{a,c\}$	mq_3 $\{a,t\}$	mq_4 $\{a,c,t\}$
$\{a,c,t\}$	$\{c,t\}$	$\{t\}$	$\{c\}$	$\emptyset$

(2.5.a) Symmetric differences between document and query models

Revision $d_1 \circ_D q$

query models → d_1 models↓	mq_1 $\{a\}$	mq_2 $\{a,c\}$	mq_3 $\{a,t\}$	mq_4 $\{a,c,t\}$
$\{a\}$	0	1	1	2
$\{a,c\}$	1	0	2	1
$\{a,t\}$	1	2	0	1
$\{a,c,t\}$	2	1	1	0

Revision $d_2 \circ_D q$

query models → d_2 models↓	mq_1 $\{a\}$	mq_2 $\{a,c\}$	mq_3 $\{a,t\}$	mq_4 $\{a,c,t\}$
$\{a,c,t\}$	2	1	1	0

(2.5.b) Cardinalities of symmetric differences

$Dist(mq_1, Mod(d_1)) = min\{0,1,1,2\} = 0$
$Dist(mq_2, Mod(d_1)) = min\{1,0,2,1\} = 0$
$Dist(mq_3, Mod(d_1)) = min\{1,2,0,1\} = 0$
$Dist(mq_4, Mod(d_1)) = min\{2,1,1,0\} = 0$

$distance(d_1, q) = \frac{\sum_{m \in Mod(q)} Dist(m, Mod(d_1))}{|Mod(q)|}$

$distance(d_1, q) = \frac{0+0+0+0}{4} = 0$

$BRsp(d_1, q) = 1 - \frac{0}{1} = 1$

$Dist(mq_1, Mod(d_2)) = min\{2\} = 2$
$Dist(mq_2, Mod(d_2)) = min\{1\} = 1$
$Dist(mq_3, Mod(d_2)) = min\{1\} = 1$
$Dist(mq_4, Mod(d_2)) = min\{0\} = 0$

$distance(d_2, q) = \frac{\sum_{m \in Mod(q)} Dist(m, Mod(d_2))}{|Mod(q)|}$

$distance(d_2, q) = \frac{2+1+1+0}{4} = 1$

$BRsp(d_2, q) = 1 - \frac{1}{3} = 0.66$

Figure 2.5: Computation of the specificity-based measure of similarity

Chapter 3

Implementation and evaluation

The main problem of logic-based IR systems is that they are complex and their implementation is difficult [7]. IR problems are intrinsically large. Therefore the tradeoff between the expressiveness of the logical representations and the complexity of the target tasks must be precisely determined. From this perspective, in this chapter we address the implementation of the model. We have studied the complexity of the tasks involved in the logical model and we provide efficient procedures for these tasks. Firstly, we analyze the complexity of the computation of the similarity measures between documents and queries represented as propositional formulas. This analysis leads to the design of efficient procedures for computing similarity. The last part of this chapter presents the experiments we have developed to evaluate our model.

3.1 Algorithms

If we try to compute $BRsim$ directly we have to get all the models of both the document and the query and compute the symmetric difference between every model of the document and every model of the query. The number of models of a propositional formula grows exponentially with the size of the propositional alphabet. Thus, a direct implementation of $BRsim$ would be useless for practical systems, which have often a large number of index terms. We propose a syntactic characterization for the formulas representing documents and queries. Specifically, a restriction in the form of the formulas allows us to design algorithms which compute the similarity between a document and a query without computing all the models of both document and query. Next paragraphs explain this characterization.

The propositional formulas which represent documents and queries have to be in disjunctive normal form (DNF). A DNF formula has the form: $c_1 \vee c_2 \vee \ldots$ where each c_j is a conjunction of literals: $l_1 \wedge l_2 \wedge \ldots$. A DNF formula can be represented by a set of clauses, $\psi = \{\psi_1, \psi_2, \ldots\}$, where each clause is a set of literals representing their conjunction. The set ψ represents the disjunction of all the clauses. For instance, the formula $(a \wedge b \wedge \neg c) \vee (c \wedge a) \vee k$ is represented by the set $\{\{a, b, \neg c\}, \{c, a\}, \{k\}\}$. A key point is that a conjunction of literals can be regarded as a partial model, representing the set of models resulting from combining the truth value of the atoms non appearing in the conjunction. For instance, given the propositional alphabet $\{a, b, c, d\}$, the conjunction $a \wedge \neg b$ represents the four models $\{a\}$, $\{a, c\}$, $\{a, d\}$ and $\{a, c, d\}$. We suggest to use a distance between clauses instead of a distance between models. A measure of distance $CDist$ between two clauses ψ_i and μ_j can be defined as follows,

Algorithm BRsim-1C:

Function Similarity(ψ,μ)
Input:
query $\psi = \{\psi_1\}$
document $\mu = \{\mu_1\}$
Output:
BRsim(μ,ψ)

1. Compute $CDist(\psi_1, \mu_1)$
2. $distance = CDist(\psi_1, \mu_1) + \frac{1}{2}(|\psi_1 \setminus \psi_1 \cap \mu_1| - CDist(\psi_1, \mu_1))$
3. return $(1 - \frac{distance}{|\psi_1|})$

Figure 3.1: Algorithm BRsim-1C

$$CDiff(\psi_i, \mu_j) = \{l \in \psi_i | \neg l \in \mu_j\} \tag{3.1}$$

$$CDist(\psi_i, \mu_j) = |CDiff(\psi_i, \mu_j)| \tag{3.2}$$

The difference between two clauses (in the following, clause difference) is given by the set of literals that appear as positive literals within one clause and as negative literals within the other clause. The distance between two clauses is the cardinality of their clause difference. From this syntactic characterization we develop now the algorithms which compute similarity between DNF formulas. The algorithms are presented in increasing order of expressiveness of the formulas representing documents and queries.

3.1.1 Algorithm BRsim-1C

The simplest situation arises when both document and query are represented as conjunctions of literals. Despite this is a restricted case, we have already pointed out in section 2.1.2 that conjunctions of literals can model classical vectors and partial vectors. A conjunction of literals is directly in DNF and can be stored as a set with a single clause. Figure 3.1 depicts the algorithm which computes the similarity measure $BRsim$ between a document and a query represented as conjunctions of literals. This algorithm is called BRsim-1C (1 clause).

The value of $CDist(\psi_1, \mu_1)$ is the number of literals in ψ_1 that appear in μ_1 with opposite value. Then, the interpretation for these terms will be different for any two models m_1 and m_2, where m_1 stands for any model of μ and m_2 stands for any model of ψ. As a result, all the models of the document have to fare at least $CDist(\psi_1, \mu_1)$ to any model of the query. So, the value of $CDist(\psi_1, \mu_1)$ is directly added to *distance*. The elements of the set $\psi_1 \setminus \psi_1 \cap \mu_1$ are the literals in ψ_1 not belonging to μ_1. Therefore, the value $|\psi_1 \setminus \psi_1 \cap \mu_1| - CDist(\psi_1, \mu_1)$ is the number of literals in ψ_1 whose letter does not appear in μ_1, either positive or negative. Since these letters do not appear in the representation of μ, half of the models of μ will map the letter into true and the other half will map it into false. On the other hand, and as a consequence of the presence of the literal in ψ_1, all the models of ψ have to map that letters into the same

truth value. Whatever this fixed truth value is, half of the models of μ will have the opposite one, increasing $\frac{1}{2}(|\psi_1 \setminus \psi_1 \cap \mu_1| - CDist(\psi_1, \mu_1))$ in distance. As a result, the value assigned to *distance* is the average distance from models of the document to the set of models of the query. Finally, the distance is transformed into a similarity value in the interval [0,1] given that the largest value of *distance* is $|\psi_1|$.

In next paragraphs we sketch why the output of this algorithm is the value of $BRsim(\mu, \psi)$. Let us recall that $BRsim$ is a normalization from a distance defined as:

$$distance(d, q) = \frac{\sum_{m \in Mod(d)} Dist(m, Mod(q))}{|Mod(d)|} \tag{3.3}$$

The value of $distance(d, q)$ is the value stored in the variable *distance* in the algorithm. In order to justify it, let us consider every propositional letter l of the alphabet. There are four possible cases:

- l appears in both d and q as a positive (negative) literal. Then, l does not contribute to the sum of the previous equation because l is mapped into true (false) by all the models of the document and by all the models of the query.

- l appears in d as a positive (negative) literal and l appears in q as a negative (positive) literal. Then, $\forall m \in Mod(d), Dist(m, Mod(q))$ is at least 1 because of l. This is because l is mapped into true (false) by all the models of the document and l is mapped into false (true) by all the models of the query. The number of these letters is precisely $CDist(\psi_1, \mu_1)$. Hence, for all the models of d the final increment to $distance(d, q)$ coming from these letters is $CDist(\psi_1, \mu_1)$.

- l does not appear in d and it appears in q. All the models of q map l into the same truth value whereas half of the models of d map l into true and half of the models of d map l into false. Then, half of the models of d add 1 to $Dist(m, Mod(q))$. The number of these letters is $|\psi_1 \setminus \psi_1 \cap \mu_1| - CDist(\psi_1, \mu_1)$. These letters contribute in $\frac{1}{2}(|\psi_1 \setminus \psi_1 \cap \mu_1| - CDist(\psi_1, \mu_1))$ to $distance(d, q)$.

- l appears in d and it does not appear in q. All the models of d map l into the same truth value whereas half of the models of q map l into true and half of the models of q map l into false. Nevertheless, since $Dist(m, Mod(q))$ takes the distance from m to the closest model in $Mod(q)$, the contribution to $distance(d, q)$ coming from these letters is equal to 0.

To sum up, it holds that:

$$distance(d, q) = CDist(\psi_1, \mu_1) + \frac{1}{2}(|\psi_1 \setminus \psi_1 \cap \mu_1| - CDist(\psi_1, \mu_1)) \tag{3.4}$$

The final normalization for computing $BRsim$ takes the number of letters appearing in q. This is precisely the value of the size of the single query clause, ψ_1. Putting all together, we can conclude that the output of the algorithm is the value of $BRsim(\mu, \psi)$.

Let us analyze the complexity of Algorithm BRsim-1C. Step 1 can be done extracting the literals of ψ_1 and checking whether the opposite literal belongs to μ_1. It can also be done with the reciprocal process, that is, extracting in μ_1 and checking in ψ_1. Each check can be done in

$\mathcal{P} = \{a, b, c, d, e\}$

$d = a \wedge b \wedge \neg e$
$q = a \wedge \neg b$

Algorithm BRsim-1C:

`Function Similarity(`ψ`,`μ`)`
`Input:`
`query` $\psi = \{\psi_1\}$, $\psi_1 = \{a, \neg b\}$
`document` $\mu = \{\mu_1\}$, $\mu_1 = \{a, b, \neg e\}$
`Output:`
`BRsim(`μ`,`ψ`)`

1. $CDist(\psi_1, \mu_1) = |\{l \in \psi_1 | \neg l \in \mu_1\}| = |\{\neg b\}| = 1$
2. $distance = 1 + \frac{1}{2}(|\{a, \neg b\} \setminus \{a\}| - 1) = 1$
3. $1 - \frac{distance}{|\psi_1|} = 1 - \frac{1}{2} = 0.5$. `return(0.5).`

Figure 3.2: Example of execution of Algorithm BRsim-1C

unit time because an array can be used to store which literals belong to a clause. Then, step 1 can be done in linear time respect to the size of either ψ_1 or μ_1. In an analogous way, step 2 can be accomplished in linear time respect to the size of either of the clauses involved. In summary, the algorithm can be executed in linear time with respect to the size of either the document or the query. Since the query has usually less literals, we can conclude that the algorithm has a complexity of $\mathcal{O}(|\psi_1|)$.

Let us consider the alphabet $\mathcal{P}$, the document d and the query q depicted in fig. 3.2. The figure shows how Algorithm BRsim-1C operates for this example. Note that the number of the steps of the algorithm is not affected by the size of the alphabet. On the contrary, a direct implementation of the computation of $BRsim$ would need to compute all the models of both document and query. For instance, if the alphabet has 100 terms the query presented below would have 2^{98} models and, as a consequence, the direct implementation of $BRsim$ is unacceptable.

An important circumstance is that a completely analogous algorithm can be designed to compute the measure of specificity, $BRsp$. In section 2.1.3 we defined $BRsp$ as:

$$distance(q, d) = \frac{\sum_{m \in Mod(q)} Dist(Mod(d), m)}{|Mod(q)|} \tag{3.5}$$

$$BRsp(d, q) = 1 - distance(q, d)/l \tag{3.6}$$

The measure is analogous to $BRsim$ but the roles of document and query are interchanged. Thus, a new version of Algorithm BRsim-1C with the inputs interchanged can compute the value of $BRsp$. Regarding complexity, we showed that Algorithm BRsim-1C can be computed in linear time with respect to the size of any of its inputs. Thus, the most efficient algorithm for computing $BRsp$ runs in linear time with respect to the size of the query.

3.1.2 Algorithm BRsim-SC

Now we consider representations of queries and documents having conjunctions and disjunctions. Queries and documents are represented by generic DNF formulas having one or more clauses. If the document has several clauses it is not possible to get the value of $BRsim$ from computations between clauses. A document clause can be used to compute the average distance from the set of its models to the query (as in step 2 of Algorithm BRsim-1C) but if two document clauses have common models, those models would be counted more that once and the average from the document models to the query would be distorted (recall that $BRsim$ needs an average of the distances from the document models to the query). As a result, we designed a new algorithm, Algorithm BRsim-SC (several clauses), which computes the set of models of the document. On the other hand, the algorithm does not need to compute the query models. We are just interested in obtaining the minimum distance from a document model to the query. The existence of common models between query clauses can just yield the existence of several minimum values but the final value of distance is not affected because we just take one of them.

The distance from a model of the document to the query is the distance from the model of the document to the nearest clause of the query. This corresponds with the semantics of the disjunction. In fact, given a disjunctive query, the satisfaction of one of the choices expressed in the disjunction is enough to consider a document as relevant, no matter how far to the document the rest of the choices are.

Next lines explain some symbols used by the algorithm. The size of the alphabet is denoted by S, i.e. $S = |\mathcal{P}|$. Given an interpretation m, $LIT(m)$ stands for the transformation of m into a set of literals, i.e. $LIT(m) = m \cup \{\neg l | l \in \mathcal{P} \setminus m\}$. The symbols ψ_{min} and ψ_{max} (μ_{max}) are the size of the smallest and the largest clause in ψ (μ), respectively.

Algorithm BRsim-SC is presented in fig. 3.3. Roughly speaking, the algorithm extracts every model of the document, computes the distance to the query, accumulates these distances and, finally, computes the average of the distances. Each model of the document is firstly transformed into a set of literals (i.e. into a clause) and then, this set is used to match against each query clause. The least distance is the distance from the model to the query and is stored in $Distance\_to\_\psi$. $Distance\_to\_\psi$ is initialized to the size of the propositional alphabet, which is an upper bound for distances to the query. Last step is the regular normalization, where the similarity measure in the interval $[0, 1]$ is obtained. The upper bound of the final distance is ψ_{min}, the size of the smallest ψ_i, because the distance from a model m of the document to the set of models of the query ψ takes the distance from m to the closest query model. For any clause ψ_i of ψ, its closest model to m fares at most the size of the clause ψ_i.

The algorithm needs at most $2^S|\mu|\mu_{max}$ iterations in order to compute the set of models of the document (step 2). The loop from step 3 to 10 is run at most 2^S times, which is the upper bound for the number of document models. The loop from step 5 to 8 takes $|\psi|$ iterations and step 6 has a worst case complexity of ψ_{max}. In summary, in the worst case the algorithm takes $2^S|\mu|\mu_{max} + 2^S|\psi|\psi_{max}$ steps. Although exponential in the size of the alphabet, a key factor is that the size of the alphabet does not depend on the input to the algorithm. Thus, S is fixed and the value 2^S is bounded by a constant.

This algorithm constitutes a substantial improvement with respect to a direct computation of $BRsim$ because no models of the query are computed. Queries have often few terms and, thus, a lot of models. An important breakthrough is achieved when query models are skipped. Although models of the document are needed, documents are expected to have few models.

Figure 3.4 depicts an example of the operation of Algorithm BRsim-SC for a document and

Algorithm BRsim-SC:

Function Similarity(ψ,μ)
Input:
query $\psi = \{\psi_1, \psi_2, \dots\}$
document $\mu = \{\mu_1, \mu_2, \dots\}$
Output:
BRsim(μ,ψ)

1. $Distance = 0; Total\_Models = 0;$
2. Compute the set of models of μ
   3. Extract a new m, model of μ
   4. $Distance\_to\_\psi = S$
      5. Extract a new $\psi_i \in \psi$
      6. Compute $CDist(\psi_i, LIT(m))$
      7. if $CDist(\psi_i, LIT(m)) < Distance\_to\_\psi$ then
         $Distance\_to\_\psi = CDist(\psi_i, LIT(m))$
      8. go to step 5 until no more ψ_is remain
   9. $Total\_Models++$; $Distance+ = Distance\_to\_\psi$
   10. go to step 3 until no more μ models remain
11. $avg\_distance = \frac{Distance}{Total\_Models}$
12. return$(1 - \frac{avg_distance}{\psi_{min}})$

Figure 3.3: Algorithm BRsim-SC

$\mathcal{P} = \{a, b, c, d\}$

$d = (a \wedge b \wedge \neg c) \vee (a \wedge \neg b \wedge c)$
$q = (a \wedge b) \vee (a \wedge c)$

Algorithm BRsim-SC:

`Function Similarity(`ψ, μ`)`
`Input:`
`query` $\psi = \{\psi_1, \psi_2\}$, $\psi_1 = \{a, b\}$, $\psi_2 = \{a, c\}$
`document` $\mu = \{\mu_1, \mu_2\}$, $\mu_1 = \{a, b, \neg c\}$, $\mu_2 = \{a, \neg b, c\}$
`Output:`
`BRsim(`μ, ψ`)`

1. $Distance = 0; Total\_Models = 0;$
2. $Mod(\mu) = \{\{a, b\}, \{a, b, d\}, \{a, c\}, \{a, c, d\}\}$
   3. $m = \{a, b\}$
   4. $Distance\_to\_\psi = 4$
      5. $\psi_1 = \{a, b\}$
      6. $LIT(m) = \{a, b, \neg c, \neg d\}$, $CDist(\psi_1, LIT(m)) = 0$
      7. $Distance\_to\_\psi = 0$
      5. $\psi_2 = \{a, c\}$
      6. $LIT(m) = \{a, b, \neg c, \neg d\}$, $CDist(\psi_2, LIT(m)) = 1$
   9. $Total\_Models = 1$; $Distance = 0$
   3. $m = \{a, b, d\}$
   4. $Distance\_to\_\psi = 4$
      5. $\psi_1 = \{a, b\}$
      6. $LIT(m) = \{a, b, \neg c, d\}$, $CDist(\psi_1, LIT(m)) = 0$
      7. $Distance\_to\_\psi = 0$
      5. $\psi_2 = \{a, c\}$
      6. $LIT(m) = \{a, b, \neg c, d\}$, $CDist(\psi_2, LIT(m)) = 1$
   9. $Total\_Models = 2$; $Distance = 0$
   3. $m = \{a, c\}$
   4. $Distance\_to\_\psi = 4$
      5. $\psi_1 = \{a, b\}$
      6. $LIT(m) = \{a, \neg b, c, \neg d\}$, $CDist(\psi_1, LIT(m)) = 1$
      7. $Distance\_to\_\psi = 1$
      5. $\psi_2 = \{a, c\}$
      6. $LIT(m) = \{a, \neg b, c, \neg d\}$, $CDist(\psi_2, LIT(m)) = 0$
      7. $Distance\_to\_\psi = 0$
   9. $Total\_Models = 3$; $Distance = 0$
   3. $m = \{a, c, d\}$
   4. $Distance\_to\_\psi = 4$
      5. $\psi_1 = \{a, b\}$
      6. $LIT(m) = \{a, \neg b, c, d\}$, $CDist(\psi_1, LIT(m)) = 1$
      7. $Distance\_to\_\psi = 1$
      5. $\psi_2 = \{a, c\}$
      6. $LIT(m) = \{a, \neg b, c, d\}$, $CDist(\psi_2, LIT(m)) = 0$
      7. $Distance\_to\_\psi = 0$
   9. $Total\_Models = 4$; $Distance = 0$
11. $avg\_distance = \frac{0}{4} = 0$
12. `return(`$1 - \frac{0}{2}$`)`

Figure 3.4: Running Algorithm BRsim-SC

Size of the alphabet	Number of doc. terms	Number of query terms	Number of doc. models	Number of query models	Run time (microsec.)
10	10	1	1	2^9	114782
10	10	2	1	2^8	68482
10	10	5	1	2^5	8233
10	10	7	1	2^3	4480
10	10	10	1	1	3756
25	25	10	1	2^{15}	781193357
25	25	13	1	2^{12}	6914517
25	25	15	1	2^{10}	1084744

Table 3.1: Run Time Performance of the models method

a query represented as DNF formulas with two clauses. Note that the final value of similarity is intuitive because every document clause satisfies at least one of the query clauses.

Algorithm BRsim-SC takes DNF formulas as an input. This means that the indexing process has to store documents in this form. However, users do not have to articulate their information needs in DNF but the translation can be done automatically by the system. This is feasible because any propositional formula can be translated into an equivalent formula in DNF. Although this translation takes exponential time, user queries have often few terms and, thus, the translation can be accomplished in reasonable time.

Again, a modified version of Algorithm BRsim-SC can be designed for computing the specificity-based measure $BRsp$. The complexity analysis of the modified algorithm is also exponential.

3.1.3 Tests

Despite having reduced the complexity with respect to a direct implementation of $BRsim$, we were wondering if the resulting algorithms, especially Algorithm BRsim-SC, were efficient enough. Therefore, we decided to run some informal tests in order to clarify the usefulness of the algorithms. We implemented Algorithm BRsim-1C and Algorithm BRsim-SC in the C programming language and a direct computation of $BRsim$ handling all the models of both document and query was also implemented (in the following we use the name of *models method* to refer to this direct approach). We used distinct alphabets with distinct sizes and we varied the size of the formulas which are the inputs to the algorithms (i.e. we varied the number of terms in the representations of the document and the query). Thus, we can analyze the behavior of the algorithms with different number of models of the input formulas.

Our tests on the models method make evident that it cannot work within realistic environments. Table 3.1 presents the run time performance of the models method. With sizes of the alphabet of more than 15 terms the algorithm took an extremely long time to compute similarity. On the other hand, Algorithm BRsim-1C showed a great performance. We tried out sizes up to 500 terms and the response time was of the order of microseconds. The results for Algorithm BRsim-1C are shown in table 3.2. As a matter of fact, the response time grew with the size of the query but it kept on the same levels when changing the alphabet size. Algorithm BRsim-1C could be used with alphabet sizes much greater than 500 and, as a result, it can stand on the basis of a realistic system.

We ran analogous tests with Algorithm BRsim-SC. The performance of Algorithm BRsim-SC was rather disappointing. Despite being better than the models method, Algorithm BRsim-SC

Size of the alphabet	Number of doc. terms	Number of query terms	Run time (microsec.)	Number of doc. models	Number of query models
25	25	1	18	1	2^{24}
25	25	2	22	1	2^{23}
25	25	5	31	1	2^{20}
25	25	7	35	1	2^{18}
25	25	10	45	1	2^{15}
25	10	1	15	2^{15}	2^{24}
25	10	2	16	2^{15}	2^{23}
25	10	5	20	2^{15}	2^{20}
25	10	7	24	2^{15}	2^{18}
25	10	10	31	2^{15}	2^{15}
50	50	1	19	1	2^{24}
50	50	2	25	1	2^{23}
50	50	5	47	1	2^{20}
50	50	7	69	1	2^{18}
50	50	10	89	1	2^{15}
50	25	1	17	2^{25}	2^{24}
50	25	2	21	2^{25}	2^{23}
50	25	5	33	2^{25}	2^{20}
50	25	7	44	2^{25}	2^{18}
50	25	10	59	2^{25}	2^{15}
50	10	1	16	2^{40}	2^{24}
50	10	2	16	2^{40}	2^{23}
50	10	5	22	2^{40}	2^{20}
50	10	7	27	2^{40}	2^{18}
50	10	10	34	2^{40}	2^{15}
500	50	1	23	2^{450}	2^{24}
500	50	2	30	2^{450}	2^{23}
500	50	5	55	2^{450}	2^{20}
500	50	7	68	2^{450}	2^{18}
500	50	10	92	2^{450}	2^{15}
500	250	1	160	2^{250}	2^{24}
500	250	2	225	2^{250}	2^{23}
500	250	5	319	2^{250}	2^{20}
500	250	7	422	2^{250}	2^{18}
500	250	10	470	2^{250}	2^{15}
500	500	1	263	1	2^{24}
500	500	2	323	1	2^{23}
500	500	5	474	1	2^{20}
500	500	7	631	1	2^{18}
500	500	10	799	1	2^{15}

Table 3.2: Run Time Performance of Algorithm BRsim-1C

Size of the alphabet	Number of doc. models	Number of query models	Run time (microsec.)
25	2^{15}	2^{20}	230280497
25	2^{10}	2^{20}	537933
25	2^{5}	2^{20}	16688
25	1	2^{20}	10470
50	2^{10}	2^{45}	12191441
50	2^{5}	2^{45}	68053
50	1	2^{45}	22975

Table 3.3: Run Time Performance of Algorithm BRsim-SC

presented only reasonable response times with alphabet sizes up to 50. Therefore, Algorithm BRsim-SC cannot be used within actual systems. Table 3.3 summarizes some results obtained when applying Algorithm BRsim-SC. To conclude, an alternative approach is needed to be able to compute similarity between general DNF formulas.

3.1.4 A new similarity measure

The source of exponentiality is the definition of the similarity measure $BRsim$ itself. The similarity measure is obtained directly from a distance that is an average of distances from document models:

$$distance(d,q) = \frac{\sum_{m \in Mod(d)} dist(Mod(q), m)}{|Mod(d)|} \tag{3.7}$$

If both document and query do not have any disjunction, their associated representations have a single clause and this clause can be used to represent the whole set of models. Algorithm BRsim-1C obtains the average of the set of models of the document without computing any model.

If the DNF formulas have more than one clause, document clauses can have common models and, as discussed above, $BRsim$ cannot be obtained without computing document models.

We propose to define a clause-based similarity measure, instead of a model-based one. Given $\psi = \{\psi_1, \psi_2, \dots\}$ and $\mu = \{\mu_1, \mu_2, \dots\}$ the DNF representations of a query and a document respectively, we propose to use the similarity measure $Csim$, defined as:

$$Cdistance(\mu,\psi) = \frac{\sum\limits_{\mu_j \in \mu} \min\limits_{\psi_i \in \psi} (CDist(\psi_i,\mu_j) + \frac{1}{2}(|\psi_i \setminus \psi_i \cap \mu_j| - CDist(\psi_i,\mu_j)))}{|\mu|} \tag{3.8}$$

$$Csim(\mu,\psi) = 1 - \frac{Cdistance(\mu,\psi)}{\psi_{min}} \tag{3.9}$$

Instead of an average over document models an average over document clauses is used. Let us recall that in the original definition of $BRsim$ an average over document models was proposed because this is the fairest choice (we do not know which interpretation is the one that corresponds

with the actual contents of the document). When handling clauses an analogous argument can be applied and, then, an average seems also to be the most appropriate decision. The distance from a document clause to the query is the distance to the nearest query clause. The distance between clauses is the number of differing terms ($CDist(\psi_j, \mu_i)$) plus half the query terms not mentioned by the document clause. The new measure is kept as close as possible to Dalal's semantics because the minimum of the distances to the clauses of the query is taken. Besides, if the formulas are DNF with a single clause then $Csim(d, q) = BRsim(d, q)$. In this sense, $Csim$ is an *approximation* to $BRsim$. Let us analyze $Csim$ from the IR perspective. The distance between a document and a query is given by an average over distances from document clauses to the query. Document clauses can be thought as different views about the document. However we do not know which view is the best (i.e. which one is the closest to the actual topics of the document). Therefore, the measure considers the average distance from document clauses to the query. To measure the distance from a document clause to the query, the model takes the distance to the closest query clause. Query clauses represent different requirements and in order to satisfy the query we just need to satisfy one of the requirements. This is why we take the closest query clause to the document clause.

There is a remarkable connection between the similarity measure $Csim$, presented above, and the matching strategies designed for the classical field of Passage Retrieval. Passage Retrieval techniques apply different methods in order to get a document divided into several parts, usually called passages. In the simplest case these passages are just paragraphs from the original document. More evolved methods use windows of text with variable size and statistical scores to determine the set of passages that represent a document. Once we have a document divided into passages, several matching strategies have been suggested in the literature. Hearst and Plaunt [27] proposed to match the query against the passages of a document and, then, the top passages are summed. The sum represents the similarity of the document with respect to the query. Callan [9] uses information from the best document passage and from the entire document to compute the similarity from a document to a query. In our similarity measure, every clause of a document can be thought as a representative of a passage of the document. Indeed, we could apply Passage Retrieval techniques to get a document divided into passages and, then, the document is represented as a DNF formula having one clause per passage. If we look again at equations 3.8 and 3.9, we can observe that our similarity measure takes into account all the document's clauses to compute the final similarity value. This means that all the document's passages would be taken into account to decide the relevance of the document to the query. We could have defined other similarity measures as Passage Retrieval works do. For instance, we could use only the similarity from the best document's clause. At this point we pretend to show that the use of expressive document's representations improves performance and we will use $Csim$ as our matching function for our experiments. The definition of new similarity measures which could outperform $Csim$ is a future line of work.

In a similar way, we can define a clause-based measure of specificity. Given $\psi = \{\psi_1, \psi_2, \dots\}$ and $\mu = \{\mu_1, \mu_2, \dots\}$ the DNF representations of a query and a document respectively, we define the similarity measure Csp as:

$$Csp\_distance(\mu, \psi) = \frac{\sum_{\psi_j \in \psi} \min_{\mu_j \in \mu} (CDist(\psi_i, \mu_j) + \frac{1}{2}(|\mu_j \setminus \psi_i \cap \mu_j| - CDist(\psi_i, \mu_j)))}{|\psi|} \quad (3.10)$$

Algorithm A_BRsim-SC:

Function Similarity(ψ,μ)
Input:
query $\psi = \{\psi_1, \psi_2, \dots\}$
document $\mu = \{\mu_1, \mu_2, \dots\}$
Output:
Csim(μ,ψ)

1. $Distance = 0$;
2. Extract a new $\mu_j \in \mu$
3. $Distance\_to\_\psi = S$
4. Extract a new $\psi_i \in \psi$
5. $d = CDist(\psi_i, \mu_j) + \frac{|\psi_i \setminus \psi_i \cap \mu_j| - CDist(\psi_i, \mu_j)}{2}$
6. if $d < Distance\_to\_\psi$ then $Distance\_to\_\psi = d$
7. go to step 4 until no more ψ_is remain
8. $Distance+ = Distance\_to\_\psi$
9. go to step 2 until no more μ_js remain
10. $avg\_distance = \frac{Distance}{|\mu|}$
11. return$(1 - \frac{avg_distance}{\psi_{min}})$

Figure 3.5: Algorithm A_BRsim-SC

$$Csp(\mu, \psi) = 1 - \frac{Csp\_distance(\mu, \psi)}{\mu_{min}} \tag{3.11}$$

The specificity-based distance between two DNF formulas is defined as an average over query clauses. The distance from a query clause to the set of document clauses is the distance to the closest document clause. The difference between two clauses is defined as above. Again, the measure Csp can be seen as an approximation to its corresponding model-based measure, $BRsp$.

3.1.5 Algorithm A_BRsim-SC

This new algorithm is a direct translation of the previous formulas into code. Figure 3.5 depicts the pseudocode of the algorithm. Every document clause (μ_j) is extracted and the distance from μ_j to the query is computed as the distance to the closest query clause (and stored in $Distance\_to\_\psi$). Final steps compute the average and the normalization. We use the name of Algorithm A_BRsim-SC to refer to this new algorithm because it computes the similarity measure $Csim$, which is an approximation to $BRsim$.

It can be easily shown that Algorithm BRsim-1C is a particular case of this new algorithm. The complexity analysis of Algorithm A_BRsim-SC is as follows. The loop from step 2 to step 9 takes $|\mu|$ iterations and the loop from step 4 to step 7 takes $|\psi|$ iterations. The computation of step 5 can be done in linear time with respect to the size of either ψ_i or μ_j. Since queries are expected to have less terms, the most efficient implementation of step 5 has a worst case

$\mathcal{P} = \{a, b, c, d, e\}$

$d = (a \wedge b \wedge d) \vee (a \wedge \neg b \wedge \neg d \wedge e)$
$q = (a \wedge e) \vee (a \wedge d)$

Algorithm A_BRsim-SC:

`Function Similarity(`ψ`,`μ`)`
`Input:`
`query` $\psi = \{\psi_1, \psi_2\}$, $\psi_1 = \{a, e\}$, $\psi_2 = \{a, d\}$
`document` $\mu = \{\mu_1, \mu_2\}$, $\mu_1 = \{a, b, d\}$, $\mu_2 = \{a, \neg b, \neg d, e\}$
`Output:`
`Csim(`μ`,`ψ`)`

1. $Distance = 0$;
   2. $\mu_1 = \{a, b, d\}$
   3. $Distance\_to\_\psi = 5$
      4. $\psi_1 = \{a, e\}$
      5. $CDist(\psi_1, \mu_1) = 0$,
         $|\psi_1 \setminus \psi_1 \cap \mu_1| = |\{a, e\} \setminus \{a\}| = 1$
         $d = 0 + \frac{1-0}{2} = 0.5$
      6. $Distance\_to\_\psi = 0.5$
      4. $\psi_2 = \{a, d\}$
      5. $CDist(\psi_2, \mu_1) = 0$,
         $|\psi_2 \setminus \psi_2 \cap \mu_1| = |\{a, d\} \setminus \{a, d\}| = 0$,
         $d = 0 + \frac{0-0}{2} = 0$
      6. $Distance\_to\_\psi = 0$
      8. $Distance = 0 + 0$;
   2. $\mu_2 = \{a, \neg b, \neg d, e\}$
   3. $Distance\_to\_\psi = 5$
      4. $\psi_1 = \{a, e\}$
      5. $CDist(\psi_1, \mu_2) = 0$,
         $|\psi_1 \setminus \psi_1 \cap \mu_2| = |\{a, e\} \setminus \{a, e\}| = 0$,
         $d = 0 + \frac{0-0}{2} = 0$
      6. $Distance\_to\_\psi = 0$
      4. $\psi_2 = \{a, d\}$
      5. $CDist(\psi_2, \mu_2) = 1$,
         $|\psi_2 \setminus \psi_2 \cap \mu_2| = |\{a, d\} \setminus \{a\}| = 1$
         $d = 1 + \frac{1-1}{2} = 1$
      6. $Distance\_to\_\psi = 0$
      8. $Distance = 0 + 0$;
10. $avg\_distance = \frac{0}{2} = 0$
11. `return(`$1 - \frac{0}{2}$`)`

Figure 3.6: Running Algorithm A_BRsim-SC

of ψ_{max} (size of the largest clause in ψ) iterations. Thus, the algorithm has a complexity of $\mathcal{O}(|\mu||\psi|\psi_{max})$. In fig. 3.6 we show an example of the process of computation of $Csim$ using Algorithm A_BRsim-SC. Note that the final value of similarity corresponds with our intuitions. The query is asking for satisfying either $(a \wedge e)$ or $(a \wedge d)$. All document clauses satisfy one of the query clauses and, thus, the document satisfies completely the query.

Again, a similar algorithm can be designed to compute the measure Csp. The complexity analysis of this algorithm leads to similar results.

Additional tests were carried out with the new algorithm. Again, we wrote C code implementing Algorithm A_BRsim-SC and the experiments were run under the same conditions as before. We tried out vocabulary sizes up to 500 terms and the response time was of the order of microseconds. Consequently, Algorithm A_BRsim-SC (and Algorithm BRsim-1C, which is a case of Algorithm A_BRsim-SC) assures a good performance within an IR system.

3.2 Usefulness of DNF formulas

We have designed procedures which compute similarity in an efficient way at the expense of fixing the syntactic form of the logical formulas involved. Formulas representing documents and queries should be in Disjunctive Normal Form. Nevertheless, the syntactic form of DNF formulas is not too limiting for IR.

DNF representations allow us to express split document's representations in an homogeneous way. Actually, the works in Passage Retrieval have shown that a system can get better performance if it represents documents divided into several parts.

We already pointed out in section 2 that areas like Image Retrieval or Speech Retrieval need document's representations more expressive than classical vectors. The output of image or speech recognition systems is inherently vague and, hence, a formalism able to model uncertainty is desirable. Specifically, DNF formulas are promising tools to face these representational problems because several alternatives about the actual contents of the documents can be modeled.

We think that the structure of a DNF formula is appropriate for users. The expressions are restricted to be in DNF and, thus, the query language is not very complex. The DNF structure divided into conjunctions (*views*) can be easily understood by users and user interfaces can be designed to help users in the articulation of DNF formulas. For instance, user interfaces should provide users with a method to sort out their interests (e.g. a user who is interested in documents dealing with either *information retrieval* or *data retrieval* could write the respective keywords into distinct graphical objects). Hence, DNF formulas can be automatically built (e.g. $(information \wedge retrieval) \vee (data \wedge retrieval)$). Furthermore, users queries can also be articulated as general Propositional Logic formulas. This is because user requests have often few terms and a translation into DNF can be automatically done.

3.3 Evaluation

In this section we present the evaluation of the model. We have built a prototype IR system implementing our model and we used it to evaluate the model against some standard benchmarks. The algorithms designed in the previous sections stand on the basis of the system. The section is organized as follows. First, we present some basic notions about IR evaluation. Then, we sketch the basic features of the prototype system. Finally, the evaluation procedures and results are presented.

3.3.1 Basic Concepts

There is a long history of experimentation in IR. Research started with experiments in indexing languages in the early sixties, and has continued with over forty years of experimentation with the retrieval engines themselves. IR evaluation has become an active research field by itself. Many efforts have been focused on the design of good evaluation methods, on the design of appropriate benchmarks and so on.

When evaluating an IR system, we should measure the factors that will reflect the ability of the system to satisfy the user. IR systems are pieces of software and, thus, they should provide the *functionality* it was conceived for. Other factors which should not be left aside are system's performance measures like time and space. A short response time and a small space used are desired features for a good system. This becomes especially important in environments like the Web where time factors are critical. Besides, other metrics are also of interest in an IR system.

Approaches to IR evaluation can be roughly divided into two main categories, namely *system-oriented evaluation* and *user-oriented evaluation*. The former measures everything which is not under control of humans, e.g. indexing approaches, weighting, query manipulation, etc. User-oriented evaluation focuses on how well users can search with systems. The ability to find information and user satisfaction are examples of user-evaluation factors. User-oriented evaluation is more difficult and much more time consuming because it implies laboratory studies with humans. On the other hand, system-oriented evaluation usually involves the use of test collections containing documents, queries and relevance judgments. Then, interaction with users is not required. A test collection is chosen and the attributes of the IR system are tested against the collection. For instance, distinct indexing or retrieval strategies can be compared to determine which approach is the best. The characteristics of the sample collections are important to determine whether we can have confidence that the results obtained can be extrapolated.

The type of evaluation to be considered depends on the objectives of the retrieval system. Although we evaluate prototype systems, we usually want to know how good is a system doing a specific task. Specific IR areas require specific evaluation methodologies. For instance, TREC conferences are divided into several tasks depending on the focus of the experimentation. This includes Filtering, Spoken Document Retrieval, Cross Language, Web, Question Answering, etc. Furthermore, specific tasks use to need variations in the evaluation procedures. For instance, in chapter 5 we will present a *residual* evaluation methodology, which is appropriate for evaluating relevance feedback approaches.

The most common evaluation method is based on testing how good is a system retrieving relevant documents. *Retrieval performance evaluation* tests how precise are the answer sets supplied by IR systems given a user query. As mentioned above, evaluations are usually done against a test reference collection, which consists of a collection of documents, a set of example information requests, and a set of relevant documents for each information request. Documents and queries are often provided in natural language and each system transforms them into an internal representation (e.g. vectors), computes similarity between the representations of queries and documents and, for each query, builds a rank of documents in decreasing order of similarity to the query. The ranks are analyzed using the relevant judgments, which are provided by specialists. In general, relevance is assumed to be binary and fixed over time and individuals. Clearly, this is a simplification because relevance is a very subjective notion. Different users may differ about the relevance or non-relevance of particular documents to given queries. Nevertheless, the difference is not large enough to invalidate experiments which have been made with document collections for which test queries with relevance assessments are available. Questions

from standard collections are often elicited from users in a particular discipline and the relevance judgments are made by a set of experts in that discipline. It is a general assumption in the field of IR that a retrieval strategy which presents a good behavior under a large number of experimental conditions then it is likely to perform well in an operational situation where relevance is not known in advance.

There are several ratios estimating the nearness from a given answer set to the ideal answer set provided by the experts. *Recall* and *Precision* are the two most used retrieval evaluation measures. Given the set R of relevant documents to a given information request and A the answer set provided by a system which implements some retrieval strategy, recall and precision are defined as follows:

$$Recall = \frac{|R_a|}{|R|} \qquad Precision = \frac{|R_a|}{|A|}, \tag{3.12}$$

where R_a is the set of relevant documents in the answer set A. Recall is the fraction of the relevant documents which has been retrieved and precision is the fraction of the retrieved documents which is relevant. Intuitively, precision-recall metrics are an attempt to measure the ability of the system to retrieve relevant documents while at the same time holding back non-relevant ones.

The output of traditional boolean systems is usually a set of documents (the ones considered relevant by the system) in no particular order. A more common situation is that systems provide answers sets ranked using a non-binary measure of similarity between documents and queries. In this case, as more documents are retrieved, recall increases while precision usually decreases. Then, a proper evaluation has to produce precision/recall values at given points in the rank. This provides an incremental view of the retrieval performance measures. The basic technique works as follows. The answer set is analyzed from the top document and the precision-recall values are computed when we find each relevant document. Figure 3.7 presents an example. The recall and precision values are computed after finding a relevant document (2nd, 5th and 8th position in the rank). Since the ranked answer set has three relevant documents, the final figure is composed of three plots. Precision/recall curves are usually based on eleven standard recall levels which are 0%,10%,...,100%. Curves like the one we obtained for the example are interpolated to get an standard curve. The interpolation process is as follows. Let $r_j, j \in \{0, 1, 2, \ldots, 10\}$, be a reference to the j-th standard recall level. The interpolated precision at the tenth standard recall level is the known precision at the recall level 100% (the non-interpolated figure always provides us with the value of precision for the recall level 100%). For the rest of the standard recall levels, the interpolated precision is the maximum known precision at any recall level between the j-th recall level and the (j+1)-th recall level.

$$P(r_j) = max_{r_j \leq r \leq r_{j+1}} P(r) \tag{3.13}$$

Note that, given an standard recall level j, if there is not known precision values (from the non-interpolated figure) between the j-th recall level and the (j+1)-th recall level, the j-th recall level inherits the interpolated precision from the (j+1)-th recall level. In fig. 3.8 we show the interpolation process for the precision/recall figure of the example depicted in fig. 3.7. At recall levels 0%, 10%, 20% and 30% the interpolated precision is equal to 0.5, which is the known precision at the recall level 33%. At recall levels 40%, 50% and 60% the interpolated precision is equal to 0.4, which is the known precision at the recall level 66%. At recall levels 70%, 80%,

Relevant documents: $R = \{d_7, d_{27}, d_{34}\}$

Rank:

1. d_{22}
2. d_7 Recall=1/3, Precision=1/2
3. d_{14}
4. d_{30}
5. d_{27} Recall=2/3, Precision=2/5
6. d_3
7. d_1
8. d_{34} Recall=3/3, Precision=3/8

Precision/Recall curve:

Recall	Precision
0.33	0.5
0.66	0.4
1	0.38

Figure 3.7: A simple recall vs. precision figure

Non-interpolated curve:

Recall	Precision
0.33	0.5
0.66	0.4
1	0.38

Interpolated curve:

Recall	Precision
0	$max_{r_0 \leq r \leq r_1} P(r) = 0.5$
0.1	$max_{r_1 \leq r \leq r_2} P(r) = 0.5$
0.2	$max_{r_2 \leq r \leq r_3} P(r) = 0.5$
0.3	$max_{r_3 \leq r \leq r_4} P(r) = 0.5$
0.4	$max_{r_4 \leq r \leq r_5} P(r) = 0.4$
0.5	$max_{r_5 \leq r \leq r_6} P(r) = 0.4$
0.6	$max_{r_6 \leq r \leq r_7} P(r) = 0.4$
0.7	$max_{r_7 \leq r \leq r_8} P(r) = 0.38$
0.8	$max_{r_8 \leq r \leq r_9} P(r) = 0.38$
0.9	$max_{r_9 \leq r \leq r_{10}} P(r) = 0.38$
1	$P(r_{10}) = P(100\%) = 0.38$

Figure 3.8: Interpolation process for a precision vs recall figure

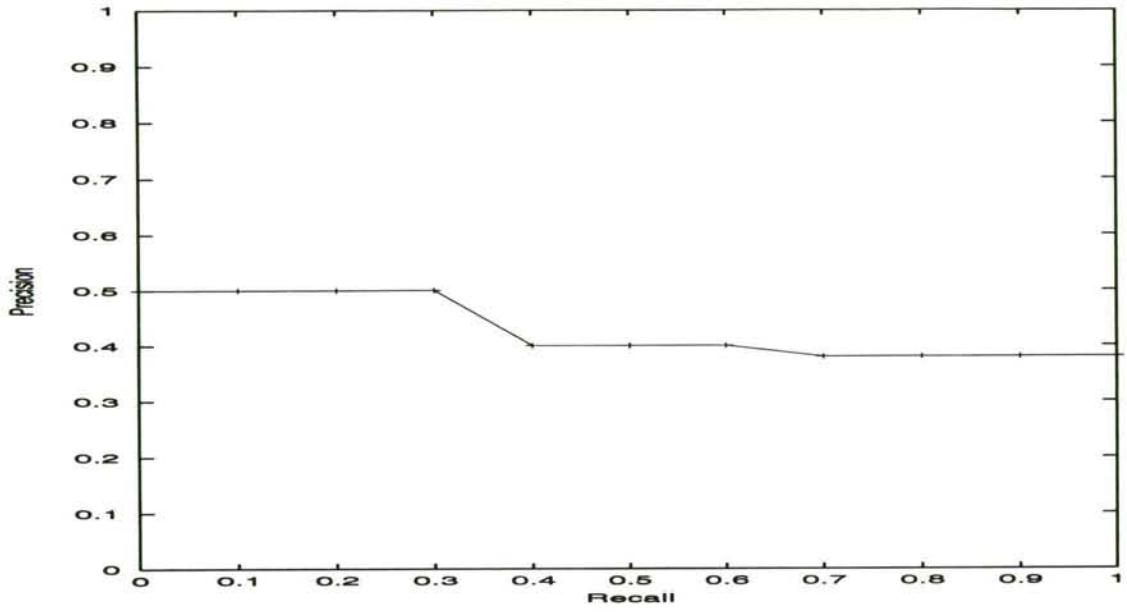

Figure 3.9: Precision vs recall graph

90% and 100% the interpolated precision is equal to 0.38, which is the known precision at the recall level 100%.

For each query a distinct precision versus recall figure is generated and to evaluate the performance of an algorithm over all test queries the precision recall figures are averaged at each recall level.

$$\overline{P}(r) = \sum_{i=1}^{N_q} \frac{P_i(r)}{N_q}, \tag{3.14}$$

where $\overline{P}(r)$ is the average precision at the recall level r, N_q is the number of queries used and $P_i(r)$ is the precision at the recall level r for the i-th query. As a result a general precision versus recall figure is obtained.

Precision versus recall figures are usually represented in a graphical way. The X axis represents values of recall and the Y axis represents values of precision. The eleven standard points are plotted and connected through lines. This allows to analyze in a visual way the evolution of precision at distinct levels of recall. Figure 3.9 depicts the precision versus recall graph for the interpolated figure shown in fig. 3.8. Such graphs are used to compare the retrieval performance of distinct retrieval algorithms in a visual way. This evaluation procedure is extensively utilized in the literature. Precision versus recall curves allow us to evaluate quantitatively both the quality of the overall answer set and the breadth of the retrieval algorithm.

An ideal system would have the highest values of both recall and precision. In practical situations precision tends to decrease as recall increases. The relative importance of precision and recall depends on the specific domain. In some domains, like the Internet, where there are many relevant documents per query, precision is more important than recall. It is more important to have a precise answer set than to retrieve all the relevant documents. Actually, given a query it is not feasible to get all its relevant documents in the Web. On the contrary, in a medical

document base which stores patient records, if a physician wants to retrieve documents about a weird disease then it is much more important to retrieve all the relevant documents (recall) than to have a precise answer set but with low recall.

Sometimes it is interesting to have a single performance value summarizing the corresponding precision versus recall figure. Along this thesis we will present values obtained for both the average precision over the eleven recall levels and the average precision for three intermediate recall levels (0.2,0.5,0.8). The former measure is simply the average of the precision values for all the recall levels.

$$\text{Avg. Prec} = \frac{\sum_{i=0}^{i=10} P(r)}{11}, \tag{3.15}$$

where $P(r)$ is the precision at the recall level r for the precision versus recall curve. The average precision for three intermediate points is analogous to the average precision over the eleven recall levels but only three recall levels are considered.

$$\text{Avg. Prec. 3 recall levels (0.2,0.5,0.8)} = \frac{P(2) + P(5) + P(8)}{3} \tag{3.16}$$

These two measures are extensively used in IR evaluation and they are appropriate in general purpose evaluation. Some other metrics have been defined in the literature. Some of them try to identify some particular characteristics of the retrieval algorithms which are hidden by the standard metrics. Nevertheless, our tests do not present any special feature which would require the use of an special measure. Besides, we computed values for some other metrics and trends did not vary. Next paragraphs sketch some other measures proposed in the area of IR evaluation.

R-Precision is defined as the precision at the R-th position in the ranking, where R is the total number of relevant documents for the current query. Again, a general R-precision value can be obtained through average over all test queries.

The *average precision at seen relevant documents* computes the average of the precision values obtained after each new relevant document is observed in the ranking. This metric favors systems which retrieve relevant documents quickly. It is feasible that a retrieval algorithm presents a good average precision at seen relevant documents but have a poor performance in terms of overall recall.

An additional approach is to compute average precision at given document cutoff values. For instance, we can compute the average precision when 5, 10, 15, 20, 30, 50 or 100 relevant documents have been seen.

Precision histograms are used to compare the retrieval history of two algorithms. The basic procedure is as follows. Let $RP_A(i)$ and $RP_B(i)$ be the R-Precision values of the retrieval algorithms A and B for the i-th query. The following distance is defined,

$$RP_{A/B}(i) = RP_A(i) - RP_B(i) \tag{3.17}$$

A value of this difference equal to 0 means that both algorithms have equivalent performance for the query. A positive value means that algorithm A is better while a negative value indicates a better performance by algorithm B. Distinct values of $RP_{A/B}(i)$ for several queries are organized in an histogram. This type of bar graph is very helpful to identify the queries for which one algorithm is better than the other through visual inspection.

Many other metrics exist in the literature and we refer to [59, 3] for a detailed description of these and other ratios.

	CACM	CRAN	CISI	LISA
Number of documents	3204	1400	1460	5999
Number of queries	52	225	76	35
Size (Kbytes)	2191	1673	2296	3914
Avg. number of relevant docs per query	16.53	8.78	47.03	14.89

Table 3.4: Size of the collections

Test collections

There are many reference collections for evaluation in IR. In this work we used four standard collections: CACM, CRAN, CISI and LISA. In these collections documents are not plain text documents but the information is divided into structured subfields. We utilized this structure to build DNF formulas for representing documents. CACM consists of articles published in the *Communications of the ACM* from the first issue in 1958 to the last number of 1979. CRAN documents are about Aerodynamics, CISI documents are Library Science articles from the Institute of Scientific Information (ISI) and LISA documents are Library and Information Science abstracts. Table 3.4 shows some data about the size of these collections.

In the early nineties, it was widely thought that IR evaluation needed consolidation. Specifically, there was two missing elements in IR evaluation. First, although some research groups had used the same collections, there had been no concerted effort by groups to work with the same data, use the same evaluation techniques, and generally compare results across systems. The importance of this is to allow comparison across a very wide variety of techniques, much wider than only one research group would tackle. The second missing element, which had become critical, was the lack of a realistically-sized test collection. Evaluation using the small collections available at that moment may not reflect performance of systems in large full-text searching, and certainly does not demonstrate any proven abilities of these systems to operate in real-world information retrieval environments. This is a major barrier to the transfer of these laboratory systems into the commercial world. Additionally some techniques such as the use of phrases and the construction of automatic thesauri seem intuitively workable, but have repeatedly failed to show improvement in performance using the small collections. Larger collections might demonstrate the effectiveness of these procedures.

The TREC project was founded under the leadership of Donna Harman at the National Institute of Standards and Technology (NIST) and attempted at addressing these two missing elements. Such an effort consisted of promoting a yearly conference, named the Text Retrieval Conference (TREC), which is dedicated to experimentation with a large test collection comprising several gigabytes. TREC is an evaluation forum in which a set of reference experiments are designed and the research groups use these experiments for comparing their retrieval systems. The TREC proceedings contain papers about tests and results of experiments against a test collections which contains approximately one million documents (about 3 gigabytes of data). To compare the results obtained there is a detailed schedule that all the participants of TREC should obey. The first TREC conference was held in November 1992 and nine TREC conferences have been taken place so far. During the years, TREC has become an standard in IR evaluation because it is the major experimental effort in the IR field.

Consequently, the big question that a researcher presenting an evaluation with a pre-TREC collection has to face is: *Why didn't you use TREC for your experiments?*. In next lines we explain why we chose four small collections instead of TREC. First, although TREC implies a

more homogeneous methodology and a good framework for comparison, it does not invalidate the evaluation which is done with old collections. In fact, many old ideas which were evaluated with old collections have become standard in the field. Furthermore, the TREC collection is a large collection which requires time consuming preparation before experiments can be carried out effectively at a local site. Besides, the testing itself is also time consuming and requires much more effort than required to execute the testing in a small collection. At this point we are not interested in making this investment. In order to test our model against TREC we would have to define efficient indexing structures to store the logical formulas representing documents and queries. On the contrary, small collections allow us to use flat structures because this does not produce an unreasonable amount of time to run the tests. Structures like inverted files could be modified to store DNF formulas in order to face a TREC evaluation. We think that this kind of system engineering tasks is out of the scope of this thesis. Then, we decided to analyze the behavior of our model through experimentation with four small collections. The evaluation against TREC has been scheduled as a future line of work. This was also the policy followed by many research groups which tested their models against small collections before testing against TREC. Nevertheless, the theoretical results obtained in section 3.1 allow us to speculate about a future TREC evaluation. Our intuition is that no major scale problems would arise with respect to the retrieval engine itself. The algorithms presented in section 3.1 are not affected by either the size of the collection or the size of the alphabet. Then, the model itself would behave well and we have just to design appropriate indexing structures to assure an efficient processing of the logical formulas.

Tests of statistical significance

Once we have two precision versus recall figures corresponding to distinct retrieval strategies, it is interesting to determine whether or not the difference between them in effectiveness is *statistically significant*. We cannot conclude that one strategy is better than the other based solely on a small performance difference between the two strategies. In fact, like any scientific experiment, IR experiments are affected by random errors.

Statistical significance should not be confused with significance from the users' point of view. A positive statistical significance test only means that the observed performance differences between two strategies are unlikely to occur by chance. Nevertheless, it can be the case that a small difference in performance is statistically significant but it has not any impact on a user. As a matter of fact, some tests of statistical significance are able to detect tiny improvements, e.g. 1% in average precision but it seems reasonable to think that these improvements would not be significant at all for a user. From the perspective of a user of an IR system, a 5% improvement in average precision is generally considered to be significant and a 10% improvement very significant.

Significance tests are used to decide whether the performance difference between two techniques is statistically significant. Now we explain the basic foundations of these tests. First, it is assumed that the two techniques being compared are equally good. Then, we compute a *p-value*, which represents the probability that the observed performance difference could occur by chance. The smaller the p-value is, the less likely that the two techniques are equally good and, thus, the more significant is the difference. A threshold is chosen and if the p-value is lower than the threshold then we are able to reject the original assumption and conclude that there is a genuine and reliable difference between the levels of performance in the two experimental conditions. Along this work, we use the value of 0.05 for this threshold. In IR, the most widely used significance tests are the *t-test* and the *sign-test*. Next paragraphs sketch the details of both

tests in a formal way.

Let us consider that we want to compare two retrieval strategies X and Y on n queries $Q_1, Q_2, \ldots, Q_n$. Since we have the performance results of each strategy, we can define n random variables as follows:

$$D_i = \text{average precision of query } Q_i \text{ using Y} - \text{average precision of query } Q_i \text{ using X} \quad (3.18)$$

T-test

The t-test assumes that $D_1, D_2, \ldots, D_n$ are n independent random variables following the same normal distribution with 0 mean and unknown variance. From this assumption, it can be easily proved that the random variable T, defined below, follows the t-distribution with $n-1$ degrees of freedom:

$$T = \frac{\overline{D}}{\frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(D_i-\overline{D})^2}}{\sqrt{n}}} \quad (3.19)$$

$$\overline{D} = \sum_{i=1}^{n} D_i \quad (3.20)$$

Hence a large value of T signifies a marked difference between the sample means and, correspondingly, a low probability that the samples vary purely by chance. Given the random variable T defined above, the p-value is:

$$\text{p-value} = 1.0 - F(n-1, |T|), \quad (3.21)$$

where F is the cumulative distribution function of the t-distribution with $n-1$ degrees of freedom.

Sign test

The sign test assumes that $P(D_i > 0) = P(D_i < 0) = 0.5$. Formally, it is assumed that the number of queries for which X is better than Y is a random variable which follows the binomial distribution with n trials and success rate 0.5. Let m be the number of queries for which X is better than Y. If n is large enough, it can be assumed that the random variable $\frac{(2m-n)}{\sqrt{n}}$ follows the standard normal distribution. Then, we can compute the p-value as follows.

$$\text{p-value} = 1.0 - G(\frac{(2m-n)}{\sqrt{n}}), \quad (3.22)$$

where G is the cumulative distribution function of the standard normal distribution.

The t-test makes an stronger assumption about the form of the distribution of the data involved but it is able to detect smaller differences between two techniques. In general, it is believed that, even when the data do not strictly follow the normal distribution, t-test results hold. Along this thesis we will use the t-test. This test is the most widely used test for IR. The sign test does not take into account the magnitude of the difference between the two variables. It only considers the number of queries in which one strategy is better than the other. In IR it is more appropriate to consider the differences in average precision as the t-test does.

3.3.2 A logic-based IR system

Now we present the prototype system that we built for evaluating the model. The basic architecture of our prototype system is drawn in fig. 3.10. The filled boxes represent elements obtained from test reference collections. The non-filled boxes represent intermediate/final elements computed by the system. Ellipses represent system's processes. First, the files containing documents and information requests are parsed, the textual information is preprocessed and DNF representations are built. Once we have DNF formulas representing documents and queries, we compute a ranked list of documents for each query. Algorithm A_BRsim-SC, presented in section 3.1.5, is in charge of the computation of the similarity between a DNF representation of a document and a DNF representation of a query. We compute the similarity between each query and each document and, for each query, a list of documents sorted in decreasing order of similarity with respect to the query is constructed. Finally, ranked lists are processed in order to get the final retrieval performance results.

Indexing

Figure 3.11 depicts a document extracted from the CACM collection and fig. 3.12 shows an information request from the same reference collection. The document is structured into several subfields, namely, title, abstract, date, authors, keywords, categories derived from a hierarchical classification scheme and some other information about co-citations between articles. The first task of the indexing procedures is to parse the documents extracting the information which is regarded as useful. In our tests we represented data from title, abstract, authors and keywords subfields. Information from all the other subfields was ignored. The preprocessing routines accomplish basic transformations on the text of each subfield. The main tasks involved in this preprocessing are:

- Lexical analysis, which treats digits, hyphens, punctuation marks, special symbols and the case of letters.
- Elimination of stopwords, which eliminates articles, prepositions, conjunctions and some other very frequent words. These words are not significant to determine the contents of a documents and, hence, they should not belong to the document's representation.
- Stemming, which is the transformation of a word into its syntactical root. Stemming techniques work on plurals, gerund forms, past tense suffixes, etc. For instance, `computer` and `computing` are both represented by `comput` and, thus, a query having the topic `computer` can match documents that do not mention `computer` but contain the term `computing`.

We used SMART's [53] list of common words for elimination of stopwords and we applied the SMART-enhanced version of the Lovins stemmer [44]. This is because we accomplished some comparisons against the SMART IR system and we wanted the comparison to be as fair as possible. The list of common words is shown in appendix B. The SMART IR system is a text processing system based on the vector space model. The primary purpose of SMART is to provide a framework in which to conduct IR research. Standard versions of indexing, retrieval and evaluation are provided. SMART automatically generates weighted vectors for any given text using several indexing schemes. We have compared the retrieval performance of our model with the retrieval performance of the vector space model with binary weights. SMART was used to index documents and queries as binary-weighted vectors, to rank documents using the

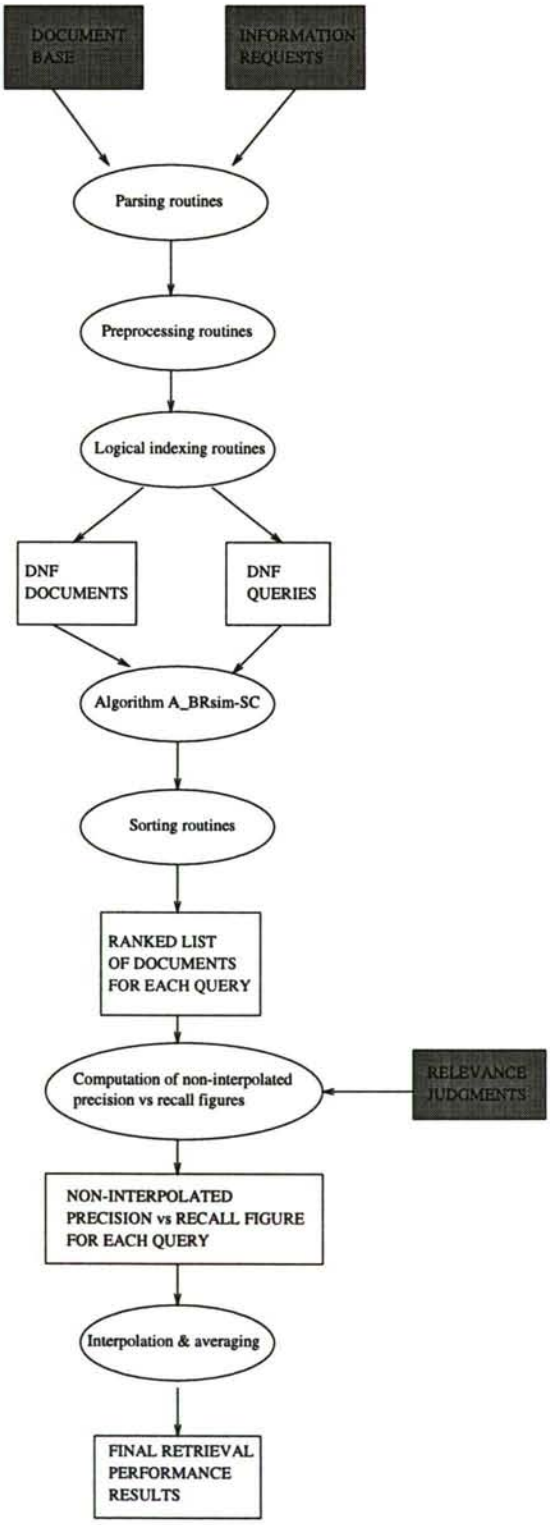

Figure 3.10: Logic-based IR system

```
.I 1664
.T
An Algorithm for Deriving the Equations of
Mathematical Physics by Symbolic Manipulation
.W
A method is described whereby a digital computer
can be used to derive the equations of mathematical
physics in any curvilinear coordinate system requested
by the user.  The effectiveness of the technique
is demonstrated by using it to derive the Navier-Stokes
equations of fluid motion and the continuity
equation.  To derive these equations by this method, the
user need know only the coordinate transformation
equations relating the curvilinear coordinates of interest
to an orthogonal Cartesian triad. When this
program is used and the coordinate transformation equations
are supplied as input, the computer will
derive the Navier-Stokes equations and the continuity
equation.  The equations obtained will be relative
to the curvilinear coordinate system specified by the
transformation equations used as input.  In this
paper the emphasis is on theoretical considerations and
methodology rather than on programming details.
 Results are presented for cylindrical polar
and spherical polar coordinate systems.
.B
CACM December, 1968
.A
Howard, J. C.
Tashjian, H.
.K
FORMAC, Navier-Stokes equations, continuity equation,
tensor, tensor equation, curvilinear coordinate
systems, FORTRAN, symbolic manipulation
.C
3.21 3.25 3.29 4.12 4.29
.N
CA681202 JB February 21, 1978  2:02 PM
.X
1543 4 1664
1543 4 1664
1664 4 1664
1664 4 1664
1664 4 1664
1345 5 1664
1543 5 1664
1616 5 1664
1664 5 1664
1664 5 1664
1664 5 1664
```

Figure 3.11: A document from CACM

```
.I 4
.W
 I'm interested in mechanisms for communicating between disjoint processes,
possibly, but not exclusively, in a distributed environment.  I would
rather see descriptions of complete mechanisms, with or without implementations,
as opposed to theoretical work on the abstract problem.  Remote procedure
calls and message-passing are examples of my interests.
.N
 4. Pavel Curtis (comm mech for disjoint processes)
```

Figure 3.12: An information request from CACM

inner product query-document matching function and to produce the final retrieval performance results.

Once the text has been preprocessed, the logical indexing routines build logical formulas for representing documents and queries. In section 3.1.5 we presented an algorithm, Algorithm A_BRsim-SC, which computes efficiently the similarity between a document and a query, both represented as DNF formulas. Consequently, in order to be able to rank documents in terms of their similarity to a given query, we have to index both documents and queries as DNF formulas. A straightforward way is to take the list of terms resulting from the preprocessing phase and construct a single clause. All the terms from all the subfields are put together and separated by logical conjunctions. Despite being simplistic, this is the usual policy in classical systems. In fact in chapter 2 we showed the correspondence between a propositional conjunctive formula and a classical vector with binary weights. Besides this simple technique, we tried out other methods for representing documents and queries. Documents in these collections are divided into distinct subfields. Intuitively, each subfield represents a different view of a document. Then, it makes sense to separate different sources of the semantics of a document into distinct clauses of the DNF representation. This leads to documents represented by DNF formulas with several clauses. This representation is inherently more expressive than a binary-weighted vector. In section 3.3.3 we compare the retrieval performance results of both approaches. Query indexing is completely analogous to document indexing.

Computing a ranked list of documents for each query

Once we have documents and queries represented as DNF formulas, a ranked list of documents for each query is computed. Given d a DNF document and q a DNF query, Algorithm A_BRsim-SC is run and the value of $Csim(d, q)$ is computed. If the specificity measure is required, Algorithm A_BRsim-SC is run again to compute $Csp(d, q)$. For each query, documents are sorted in decreasing order of similarity.

Evaluation procedures

The final tasks depicted in fig. 3.10 correspond to the evaluation procedures which are needed to compute the final retrieval performance results. For each query we take its ranked list of documents and a non-interpolated precision-recall figure is computed. As in the example depicted in fig. 3.7, the ranked lists are analyzed from the top document and the values of precision are computed after finding each relevant document. The relevance judgments of the associated collection provide us with the set of relevant documents for each query. Finally, each precision

	CACM	CRAN	CISI	LISA
Title	x	x	x	x
Abstract	x	x	x	x
Authors	x	x	x	
Keywords	x		x	

Table 3.5: Document subfield structure in the CACM, CRAN, CISI and LISA collections

versus recall figure is interpolated and, thus, we have a set of precision versus recall figures with the standard eleven recall levels. These figures are averaged over all queries at each recall level. This leads to a general precision versus recall figure that summarizes the retrieval performance of each run.

3.3.3 Evaluation results

Four classical test reference collections have been used for evaluating our model: CACM, CRAN, CISI and LISA. For all the collections, the indexing procedures only took into account the subfields title, abstract, authors and keywords, if available. Not all these subfields exist within all the collections. Table 3.5 shows which subfields are present in documents from each collection. Next sections present the details of the experiments that we ran. First, we tested the impact of expressive representations on retrieval performance. We tried out DNF formulas with a single clause and generic DNF formulas with several clauses. The latter formulas are more expressive because both logical conjunctions and logical disjunctions are handled. The last experiments presented in this section aimed at rating the impact of the measure of specificity on performance. Instead of using $Csim$ to compute similarity (i.e. only exhaustivity) a measure combining both exhaustivity and specificity is applied.

DNF formulas with a single clause

First, we ran experiments with documents and queries represented as DNF formulas with a single clause. This aims at having a baseline for later experiments applying more ellaborated representations. Besides, this first pool of experiments allowed us to study which document subfields should be considered in each collection. Intuitively, the more data we use, the better the relevance decision will be. Anyway we wanted to check the impact of individual subfields on system performance. We also wanted to assure that our model with DNF representations having a single clause gives similar results than the vector space model with binary weights. Hence, we used the SMART system to produce performance results for the vector space model with binary weights. We already showed in section 2.1.3 that our model can represent binary-weighted vectors as conjunctions of literals and we also demonstrated that the similarity measure $BRsim$ subsumes the inner product query-document matching function. However, we wanted to show this equivalence in an empirical setting.

Figure 3.13 presents an example of the indexing process applied in these experiments. Basically, the text from the subfields of the document is preprocessed and then a conjunctive formula is built. Note that the document was extracted from the LISA collection and, hence, it has only subfields for title and abstract (see table 3.5). Indexing of queries works in the same way. Once documents and queries were properly indexed, our prototype system provided us with retrieval performance results for the four collections.

Document 12
TOWARDS A REDEFINITION OF BIBLIOGRAPHY.

TRACES THE HISTORY OF THE USAGE OF THE TERM BIBLIOGRAPHY, AND ARGUES FOR A WIDER DEFINITION BASED ON THE ANGLO-AMERICAN CONCEPT OF 'REFERENCE SERVICE' TO ACCOMMODATE THE DIVERSITY OF DOCUMENTS AND MODES OF ACCESS NOW AVAILABLE.

Logical Representation:

$redefin \wedge bibliograph \wedge trac \wedge hist \wedge usag \wedge term \wedge argu \wedge wid \wedge defin \wedge base \wedge angl \wedge americ \wedge concept \wedge refer \wedge servic \wedge accommod \wedge divers \wedge docu \wedge mode \wedge access$

Figure 3.13: A LISA document and its logical representation

SMART experiments were run under the following conditions. A binary weighting scheme (for those who are familiar with SMART, we used `bnn` for the fields `doc_weight` and `query_weight`) was used to create query and document vectors. All the terms not appearing within a document/query are assigned weight 0 in the document/query vector, whereas the terms that appear within a document/query receive weight 1. Rankings are built in decreasing order of the inner-product between query and document vectors. In the following we will use the name BVSP to refer to the vector space model with binary weights tested with SMART and the name PLBRM (Propositional Logic Belief Revision Model) to refer to our model. At this point we do not pretend to compare the performance of BVSP with the performance of PLBRM. As a matter of fact, in section 2.1.3 we already demonstrated that our similarity measure $BRsim$ subsumes the inner product query-document matching function. The equivalence is obtained when the logical formulas are built under a close world assumption (CWA). For instance, in fig. 3.13, a CWA policy would assume that the formula representing the document includes the negation of all the terms not appearing in the document. However, in all our tests we take benefit from the ability of the logic to deal with uncertainty and, then, we write formulas like the one in the figure, where it is assumed that we do not know whether the document is (or is not) actually about the missing terms. We do not expect to get significant improvements from the use of partial vectors (conjunctions without CWA in PLBRM) with respect to the use of total vectors (BVSP). The difference between both approaches stands on the treatment of the query terms that do not appear within the document. In such a case, a weight 0 is assigned to the term in the document vector. On the contrary, we do not store information about that term in the logical representation of the document. When computing similarity, the term increases 0.5 the final value of distance from the document to the query (because half of the models of the formula map the term into true and half of the models map the term into false). If the document actually deals with the concept represented by the term, the assignment of a 0-weight for the term in the document's representation makes a mistake. On the contrary, if the document is not about the concept represented by the term, a 0-weight is completely accurate. For any of the two cases, our approach fares 0.5 to the good decision. Thus, our intuition is that, on average, both approaches should produce similar performance results.

Tables 3.6, 3.7, 3.8 and 3.9 present the precision/recall values for the first pool of experiments. The first columns of each table depict the precision/recall values obtained from our system and

Recall - Precision	PLBRM					BVSP
	T	W	TW	TWK	TWKA	TWKA
0.00	0.4253	0.5086	0.5281	0.5566	0.5570	0.5444
0.10	0.3054	0.3769	0.3977	0.4422	0.4468	0.4594
0.20	0.2272	0.2546	0.2828	0.3280	0.3295	0.3495
0.30	0.1635	0.1751	0.1887	0.2336	0.2422	0.2733
0.40	0.1090	0.1265	0.1649	0.1896	0.1916	0.2081
0.50	0.0834	0.1123	0.1504	0.1586	0.1598	0.1803
0.60	0.0636	0.0862	0.1265	0.1286	0.1299	0.1525
0.70	0.0495	0.0469	0.0661	0.0776	0.0795	0.0830
0.80	0.0424	0.0401	0.0536	0.0668	0.0679	0.0610
0.90	0.0307	0.0302	0.0357	0.0420	0.0419	0.0415
1.00	0.0294	0.0292	0.0304	0.0337	0.0337	0.0302
Avg. prec.	0.1390	0.1624	0.1841	0.2052	0.2072	0.2167
% Prec. change		+16.8 %	+32.4 %	+47.6 %	+49.1 %	
Avg. prec. for 3 intermediate points	0.1177	0.1357	0.1623	0.1845	0.1857	0.1969
% Prec. change		+15.3 %	+37.9 %	+56.8 %	+57.8 %	

Table 3.6: CACM

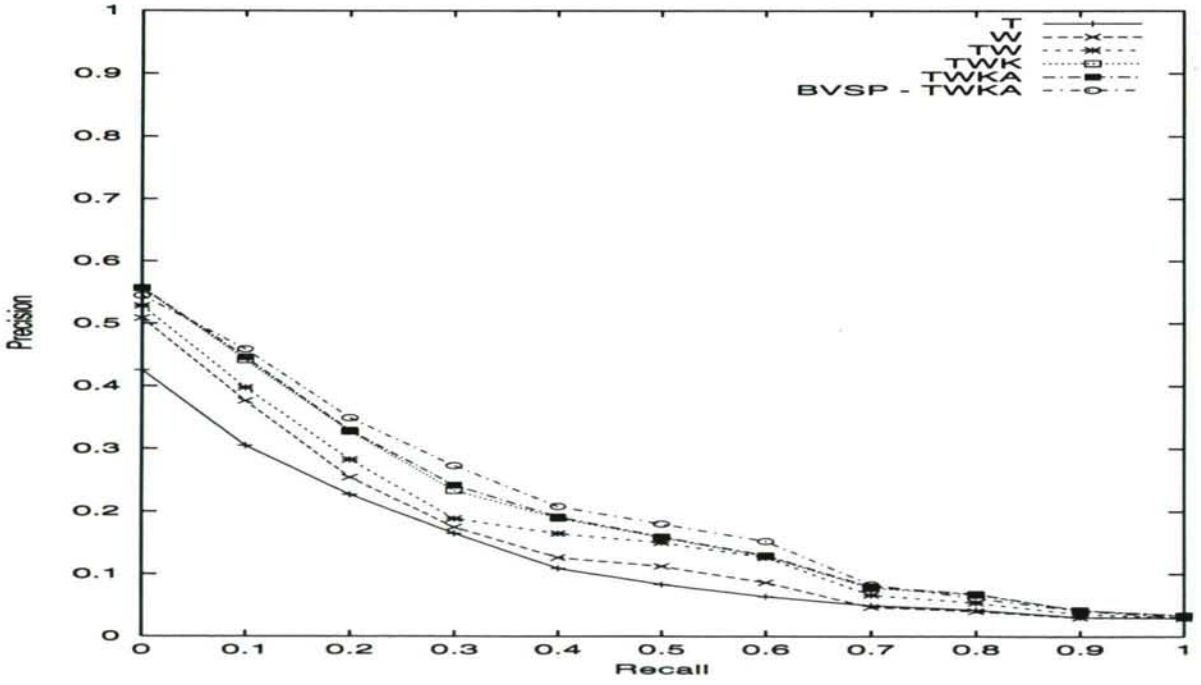

Figure 3.14: CACM - Precision vs recall graph

Recall - Precision	PLBRM				BVSP
	T	W	TW	TWA	TWA
0.00	0.6946	0.6494	0.6494	0.6495	0.6369
0.10	0.6542	0.6086	0.6086	0.6087	0.5855
0.20	0.5340	0.4835	0.4835	0.4836	0.4566
0.30	0.4118	0.3558	0.3558	0.3559	0.3454
0.40	0.3291	0.2857	0.2857	0.2859	0.2742
0.50	0.2866	0.2529	0.2529	0.2532	0.2344
0.60	0.2131	0.1895	0.1895	0.1895	0.1848
0.70	0.1372	0.1192	0.1192	0.1192	0.1198
0.80	0.1025	0.0937	0.0937	0.0937	0.0915
0.90	0.0672	0.0678	0.0668	0.0678	0.0679
1.00	0.0654	0.0646	0.0646	0.0647	0.0623
Avg. prec.	0.3178	0.2882	0.2882	0.2883	0.2781
% Prec. change		-9.3 %	-9.3 %	-9.3 %	
Avg. prec. for 3 intermediate points	0.3077	0.2767	0.2767	0.2768	0.2608
% Prec. change		-10.1 %	-10.1 %	-10.0 %	

Table 3.7: CRAN

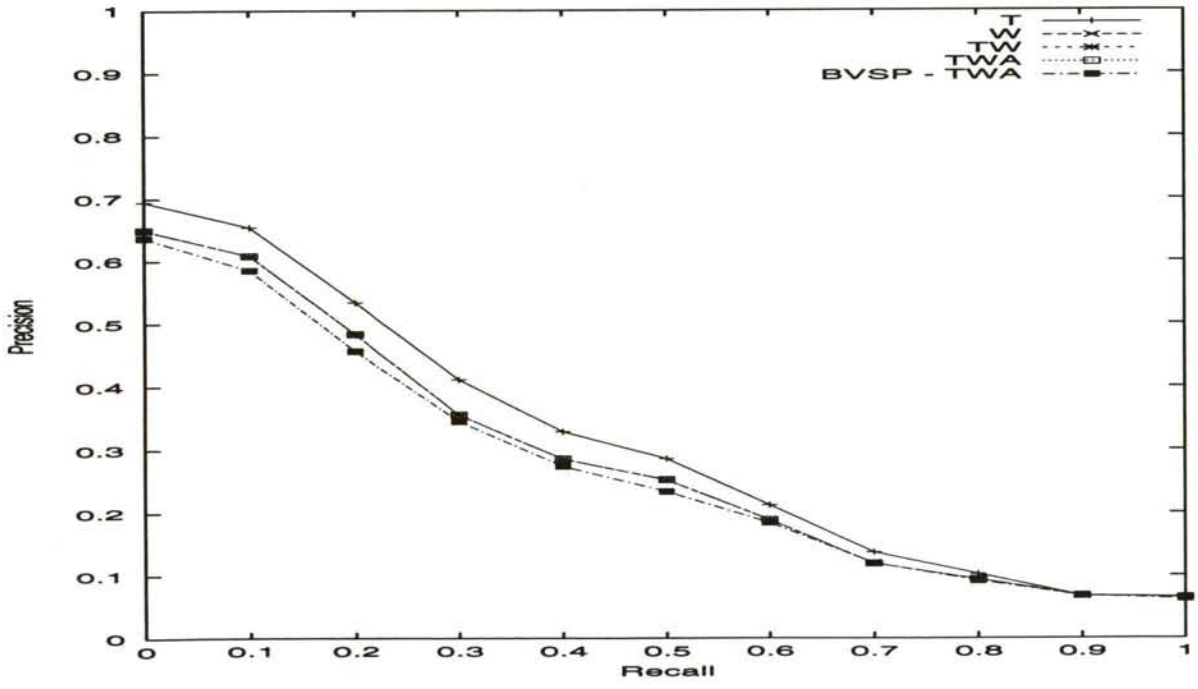

Figure 3.15: CRAN - Precision vs recall graph

Recall - Precision	PLBRM					BVSP
	T	W	TW	TWK	TWKA	TWKA
0.00	0.3982	0.4667	0.4808	0.4807	0.4826	0.4528
0.10	0.1873	0.2453	0.2459	0.2459	0.2461	0.2225
0.20	0.1316	0.1753	0.1773	0.1773	0.1757	0.1648
0.30	0.1136	0.1427	0.1438	0.1438	0.1451	0.1258
0.40	0.0876	0.1219	0.1241	0.1241	0.1249	0.1090
0.50	0.0765	0.1034	0.1065	0.1065	0.1076	0.0911
0.60	0.0636	0.0892	0.0935	0.0934	0.0936	0.0789
0.70	0.0524	0.0780	0.0801	0.0801	0.0803	0.0677
0.80	0.0453	0.0676	0.0710	0.0710	0.0711	0.0593
0.90	0.0372	0.0575	0.0596	0.0596	0.0597	0.0474
1.00	0.0324	0.0473	0.0486	0.0486	0.0485	0.0366
Avg. prec.	0.1114	0.1450	0.1483	0.1483	0.1487	0.1324
% Prec. change		+30.2 %	+33.1 %	+33.1 %	+33.5 %	
Avg. prec. for 3 intermediate points	0.0845	0.1154	0.1182	0.1182	0.1181	0.1051
% Prec. change		+36.6 %	+39.9 %	+39.9 %	+39.8 %	

Table 3.8: CISI

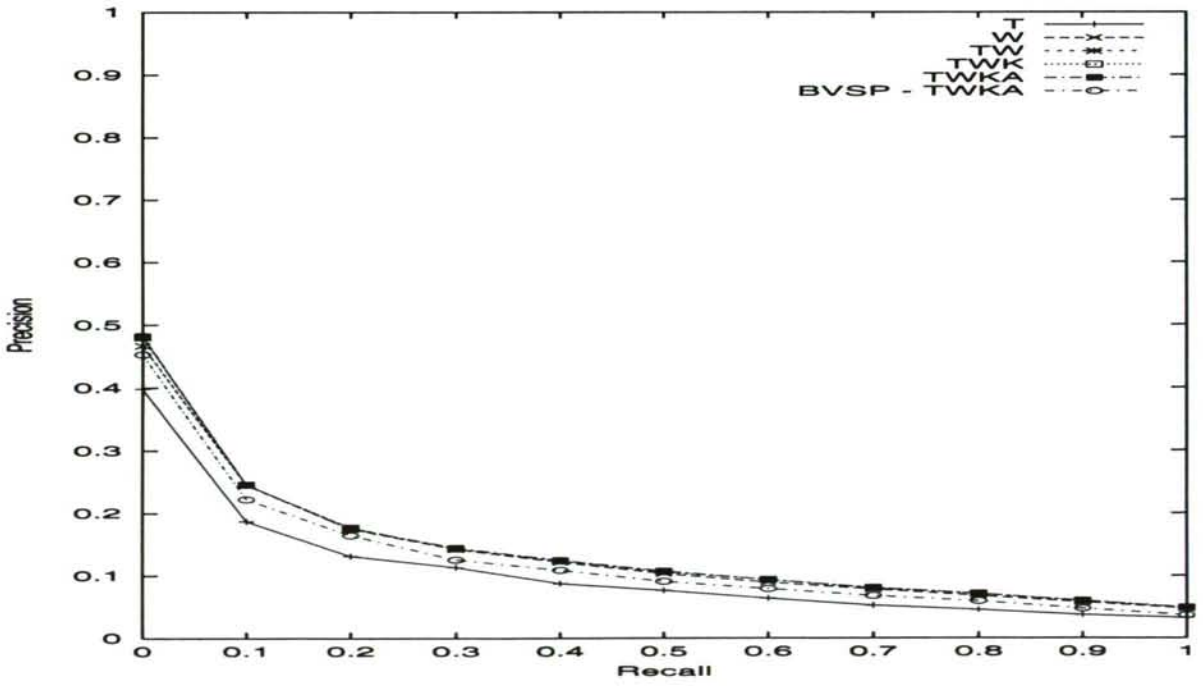

Figure 3.16: CISI - Precision vs recall graph

Recall - Precision	PLBRM			BVSP
	T	W	TW	TW
0.00	0.1820	0.1932	0.2095	0.1801
0.10	0.1081	0.1462	0.1760	0.1137
0.20	0.0620	0.1019	0.1295	0.0816
0.30	0.0424	0.0586	0.0859	0.0524
0.40	0.0290	0.0207	0.0256	0.0142
0.50	0.0107	0.0114	0.0121	0.0095
0.60	0.0073	0.0054	0.0062	0.0072
0.70	0.0038	0.0041	0.0047	0.0045
0.80	0.0032	0.0037	0.0041	0.0040
0.90	0.0029	0.0030	0.0035	0.0032
1.00	0.0026	0.0026	0.0031	0.0027
Avg. prec.	0.0413	0.0501	0.0600	0.0430
% Prec. change		+21.3 %	+45.3 %	
Avg. prec. for 3 intermediate points	0.0253	0.0390	0.0485	0.0317
% Prec. change		+54.1 %	+91.7 %	

Table 3.9: LISA

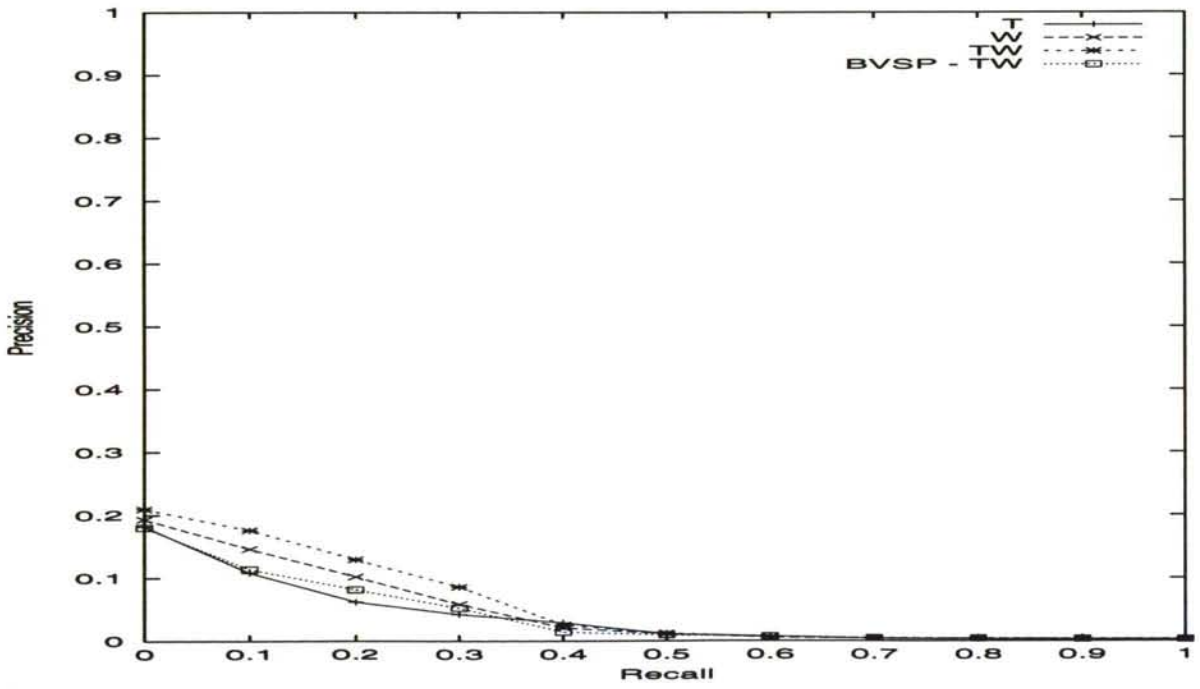

Figure 3.17: LISA - Precision vs recall graph

	CACM	CRAN	CISI	LISA
Avg. num. of terms in Title	5.27	7.88	5.21	5.87
Avg. num. of terms in Abstract	48.36	86.10	58.69	43.10

Table 3.10: Average number of terms in title and abstract

the last column of each table presents results for the BVSP obtained from SMART. In each run a different set of document subfields was used. In the tables, T stands for the title subfield, W for the abstract subfield, K for the keywords subfield and A for the authors subfield. On the top of each column we show the letters corresponding to the document subfields that were indexed. Figures 3.14, 3.15, 3.16 and 3.17 illustrate the corresponding precision versus recall figures from a graphical point of view. In general, performance ratios improved as more subfields were considered. A surprising circumstance is that CRAN's best results are obtained when only the title subfield is indexed. Consider table 3.10, which depicts the average number of terms (after stopword processing and stemming) within the title and abstract subfields in each collection. CRAN's abstracts are large and they likely contain many terms which are not important to determine the actual contents of the documents. If we merge abstracts and titles in a single clause, important and not important terms are mixed and the resulting representation can be imprecise. In fact, weights in our model are binary and, thus, term frequency information cannot be used to measure the relative importance of distinct keywords. On the other hand, CRAN's titles are not particularly short and, hence, they are on their own good representatives for documents. Anyway, we will show later that, even in the CRAN collection, the use information from all the subfields produces significant improvements if more expressive logical formulas are constructed.

In each collection the BVSP was tested using the highest number of subfields available. If we compare the BVSP run with its corresponding run in our system we can observe that in three out of the four collections the run in our system was better. Only in CACM BVSP outperformed PLBRM. Anyway, the difference is quite small for all the collections. This confirms our previous intuitions because BSVP's performance is roughly the same as the performance of PLBRM when only conjunctive representations are used.

We have empirically demonstrated that our logical model works well with some classical collections and, further, the classical BVSP does not outperform our model. To conclude, we can state that PLBRM can constitute the basis of a working IR system with retrieval performance results similar to the ones supplied by the BVSP model.

Those who are not familiar with IR test collections might wonder why retrieval performance results from different collections are so different. IR test collections are intrinsically different. The subjects of documents and queries are not the same, the level of homogeneity of the vocabulary is different, the number of relevant documents per query is different, the queries are not equally good in all the collections and so forth. All these factors lead to retrieval performance results which can be very different in distinct collections. The objective is not to analyze a given retrieval strategy in several collections from an absolute point of view but to study for each collection the variations in performance when applying distinct retrieval strategies. If the same tendency is observed in all the collections then a conclusive hypothesis can be articulated. It is not sensible to compare the retrieval performance results of a retrieval strategy for a collection A with the retrieval performance results of the same retrieval strategy for a collection B because A and B are likely very different to each other.

DNF formulas with several clauses

In the second pool of experiments that we run, generic DNF formulas, having conjunctions and disjunctions, were used for indexing documents and queries. The aim of these tests was to determine whether more expressive representations benefit the IR system in terms of precision and recall. To obtain a DNF representation for a document, terms from distinct subfields are separated into distinct clauses. Intuitively, different subfields represent aspects of the document which are semantically different and, thus, it makes sense to isolate each part in a clause. Since DNF formulas provide us with a method to articulate several views of a document, we use each view to express a subfield. Figure 3.18 depicts the same document of the fig. 3.13 but now the document is indexed by a DNF formula with several clauses. Regarding queries, only the CISI collection has a query subfield structure which is complex enough to be able to articulate DNF formulas with several clauses. In all the other collections queries have often a single subfield. As a result, queries in CACM, CRAN and LISA were indexed as DNF formulas with a single clause, as in the previous tests.

Document 12
TOWARDS A REDEFINITION OF BIBLIOGRAPHY.

TRACES THE HISTORY OF THE USAGE OF THE TERM BIBLIOGRAPHY, AND ARGUES FOR A WIDER DEFINITION BASED ON THE ANGLO-AMERICAN CONCEPT OF 'REFERENCE SERVICE' TO ACCOMMODATE THE DIVERSITY OF DOCUMENTS AND MODES OF ACCESS NOW AVAILABLE.

Logical Representation:

$(redefin \wedge bibliograph) \vee (trac \wedge hist \wedge usag \wedge term \wedge bibliograph \wedge argu \wedge wid \wedge defin \wedge base \wedge angl \wedge americ \wedge concept \wedge refer \wedge servic \wedge accommod \wedge divers \wedge docu \wedge mode \wedge access)$

Figure 3.18: A LISA document and its logical representation

Tables 3.11, 3.12, 3.13 and 3.14 summarize the evaluation results obtained after applying the new indexing technique. In the following we will use the name of *AND-tests* to refer to the previous tests, when both documents and queries were represented by DNF formulas with a single clause. By *AND/OR-tests* we refer to the new tests presented here, with generic DNF formulas. In each table we show again the results for the best AND-test. The results from SMART for the BVSP are also repeated here. For the CRAN collection we repeat not only the best AND-test (indexing only title, T) but the TWA AND-test is also repeated. This is because the AND/OR-test on the CRAN collection indexes the three subfields T, W and A and, thus, we want to compare it with its corresponding AND-test. Again, for each table, we present its corresponding graphical figure (figs. 3.19, 3.20, 3.21 and 3.22).

In the CACM collection, the AND/OR-test outperforms clearly its corresponding AND-test (22.1% in terms of average precision). The test on BVSP, despite being slightly better than the best AND-test (4.6% in terms of average precision), is significantly inferior to the AND/OR-test.

For the CRAN collection similar results are obtained. The AND/OR test improves significantly the best AND-test, which indexed only the title subfield (15.8% in terms of average

Recall - Precision	PLBRM, AND-test	BVSP	PLBRM, AND/OR-test
	TWKA Only and's in both d & q	TWKA BVSP	TWKA And's & Or's in d Only and's in q
0.00	0.5570	0.5444	0.6004
0.10	0.4468	0.4594	0.4988
0.20	0.3295	0.3495	0.3935
0.30	0.2422	0.2733	0.3227
0.40	0.1916	0.2081	0.2642
0.50	0.1598	0.1803	0.2374
0.60	0.1299	0.1525	0.1739
0.70	0.0795	0.0830	0.1085
0.80	0.0679	0.0610	0.0925
0.90	0.0419	0.0415	0.0513
1.00	0.0337	0.0302	0.0393
Avg. prec. % Prec. change	0.2072	0.2167 +4.6%	0.2530 +22.1%
Avg. prec. for 3 intermediate points % Prec. change	0.1857	0.1969 +6%	0.2412 +29.9%

Table 3.11: CACM

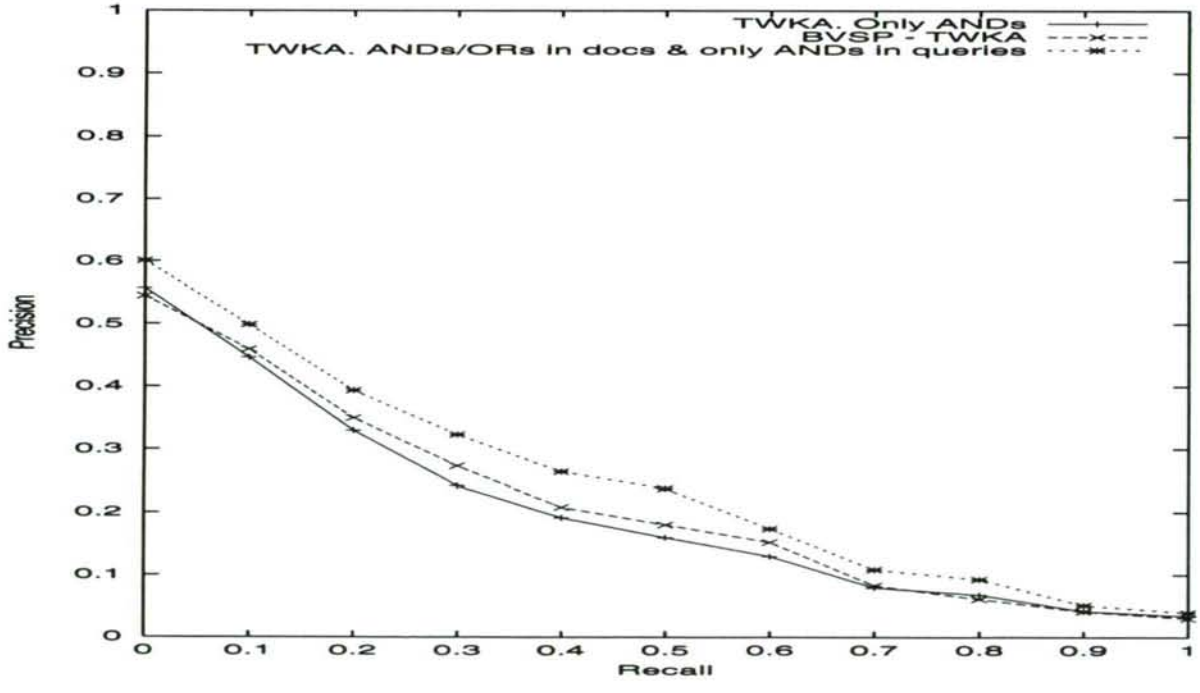

Figure 3.19: CACM - Precision vs recall graph

Recall - Precision	PLBRM, AND-tests		BVSP	PLBRM, AND/OR-test
	T Only and's in both d & q	TWA Only and's in both d & q	TWA	TWA And's & Or's in d Only and's in q
0.00	0.6946	0.6495	0.6369	0.7430
0.10	0.6542	0.6087	0.5855	0.6968
0.20	0.5340	0.4836	0.4566	0.5752
0.30	0.4118	0.3559	0.3454	0.4860
0.40	0.3291	0.2859	0.2742	0.4165
0.50	0.2866	0.2532	0.2344	0.3562
0.60	0.2131	0.1895	0.1848	0.2766
0.70	0.1372	0.1192	0.1198	0.1777
0.80	0.1025	0.0937	0.0915	0.1358
0.90	0.0672	0.0678	0.0679	0.0949
1.00	0.0654	0.0647	0.0623	0.0897
Avg. prec. % Prec. change	0.3178	0.2883 -9.3 %	0.2781 -12.5 %	0.3680 +15.8%
Avg. prec. for 3 intermediate points % Prec. change	0.3077	0.2768 -10.0 %	0.2608 -15.2 %	0.3478 +13.0%

Table 3.12: CRAN

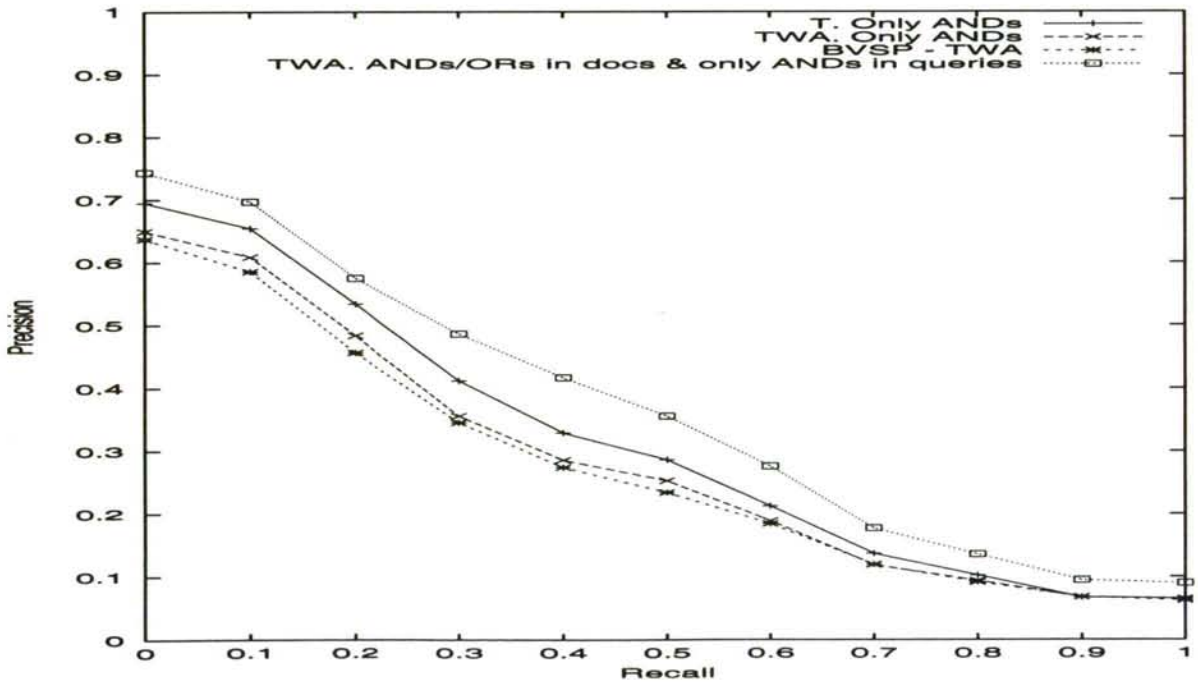

Figure 3.20: CRAN - Precision vs recall graph

Precision Recall	PLBRM AND-test	BVSP	PLBRM AND/OR-tests		
	TWKA Only and's in both d & q	TWKA	TWKA And's & Or's in d Only and's in q	TWKA And's & Or's in both d & q	TWKA Only and's in d And's & Or's in q
0.00	0.4826	0.4528	0.4751	0.4873	0.5308
0.10	0.2461	0.2225	0.2617	0.2851	0.2865
0.20	0.1757	0.1648	0.1880	0.1971	0.1997
0.30	0.1451	0.1258	0.1471	0.1490	0.1607
0.40	0.1249	0.1090	0.1235	0.1257	0.1310
0.50	0.1076	0.0911	0.1046	0.0989	0.1080
0.60	0.0936	0.0789	0.0895	0.0843	0.0924
0.70	0.0803	0.0677	0.0740	0.0700	0.0766
0.80	0.0711	0.0593	0.0634	0.0597	0.0671
0.90	0.0597	0.0474	0.0515	0.0485	0.0571
1.00	0.0485	0.0366	0.0399	0.0386	0.0476
Avg. prec.	0.1487	0.1324	0.1471	0.1495	0.1598
% Prec.change		-11.0%	-1.1%	+0.5%	+7.5%
Avg. prec. 3 int. points	0.1181	0.1051	0.1187	0.1185	0.1250
% Prec. change		-11.0%	+0.5%	+0.3%	+5.8%

Table 3.13: CISI

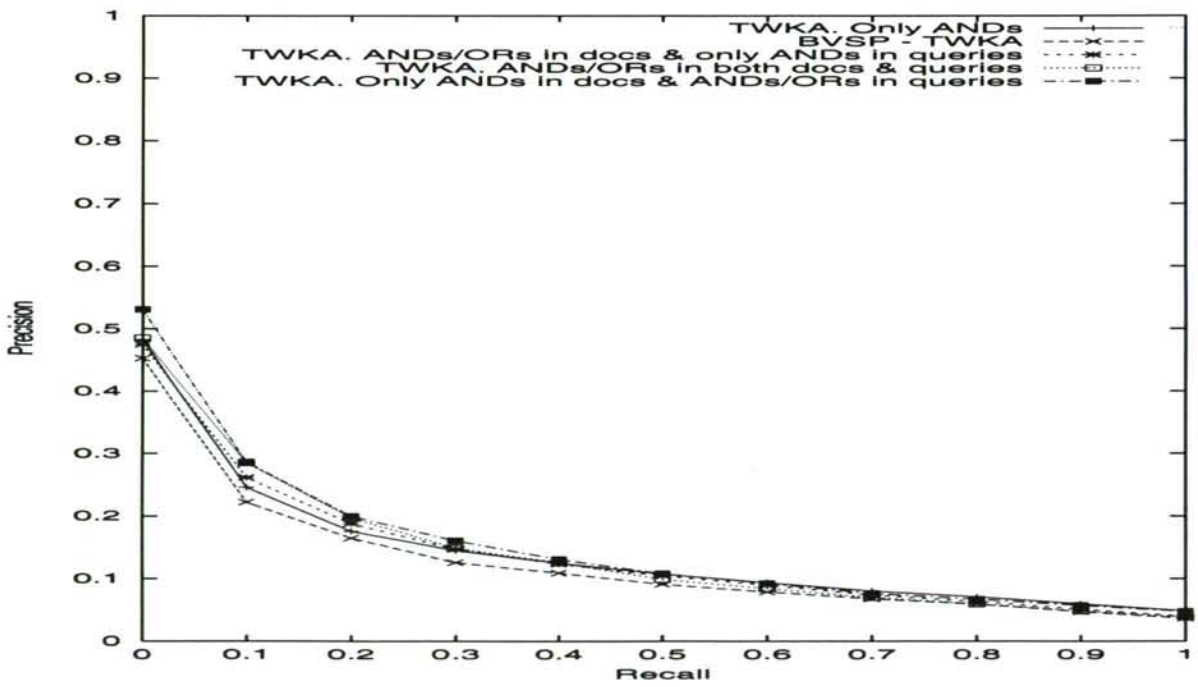

Figure 3.21: CISI - Precision vs recall graph

Recall - Precision	PLBRM, AND-test	BVSP	PLBRM, AND/OR-test
	TW Only and's in both d & q	TW	TW And's & Or's in d Only and's in q
0.00	0.2095	0.1801	0.2332
0.10	0.1760	0.1137	0.1816
0.20	0.1295	0.0816	0.1332
0.30	0.0859	0.0524	0.0888
0.40	0.0256	0.0142	0.0350
0.50	0.0121	0.0095	0.0158
0.60	0.0062	0.0072	0.0069
0.70	0.0047	0.0045	0.0054
0.80	0.0041	0.0040	0.0044
0.90	0.0035	0.0032	0.0038
1.00	0.0031	0.0027	0.0034
Avg. prec. % Prec. change	0.0600	0.0430 -28.3%	0.0647 +7.8%
Avg. prec. for 3 intermediate points % Prec. change	0.0485	0.0317 -3.5%	0.0512 +5.6%

Table 3.14: LISA

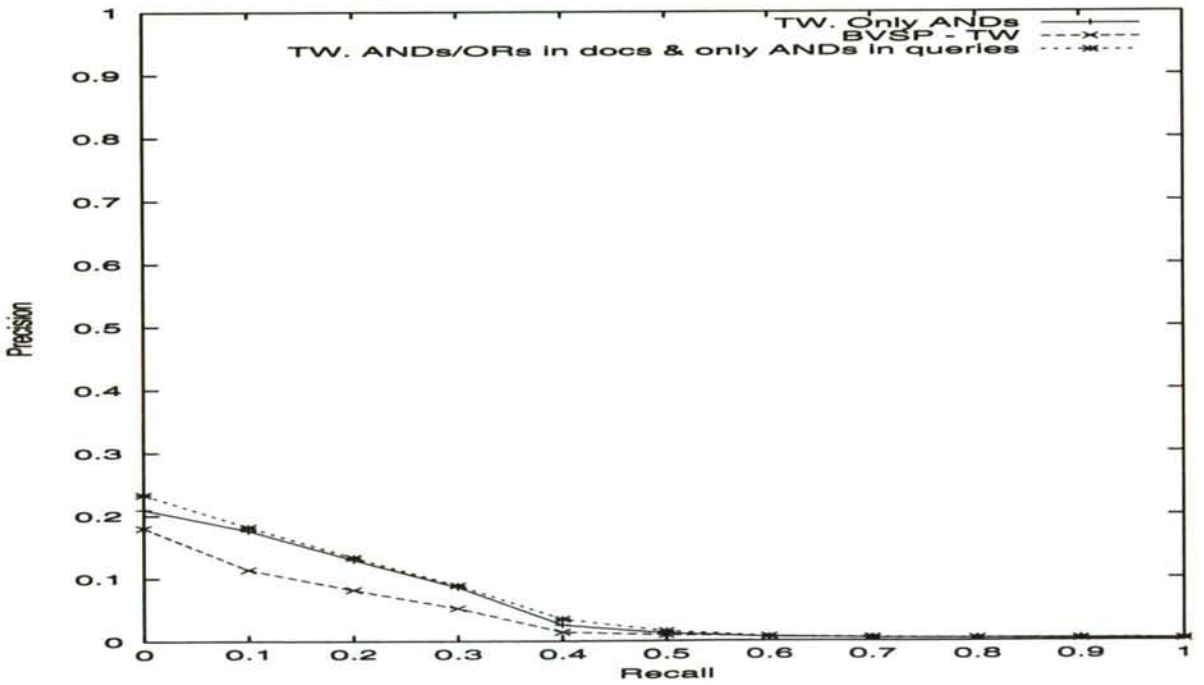

Figure 3.22: LISA - Precision vs recall graph

precision). Besides, the new test is quite better than both its corresponding AND-test (which indexes the same subfields namely T, W and A) and the test on BVSP. This shows that the use of a proper formalism allows to include efficiently (in terms of retrieval performance) information from all the subfields. On the contrary, the previous AND-test showed that a binary-weighted vector is not appropriate for integrating the information from all the subfields.

The differences between the AND-tests and the AND/OR tests are narrower for the CISI and LISA collections. For the CISI collection we could express DNF formulas with several clauses for both documents and queries. Unlike in the other collections, the AND/OR test with documents indexed as generic DNF formulas and queries indexed as DNF formulas with a single clause did not outperform the corresponding AND-test. If we look again at the AND-tests for CISI (table 3.8), we note that the results for the best AND-test (TWKA) are quite similar to the ones obtained for the W AND-test (0.1487 versus 0.1450 in average precision, i.e. only a difference of 2.5%). This means that the biggest influence on performance comes from the subfield W (abstract), whereas the influence from other subfields is minimum. Then, even applying more expressive formulas for separating distinct subfields we do not get improvements in performance. The differences from the W AND-test to the best AND-test are much larger in the other collections. Specifically, the differences in average precision between the best AND-test and the W AND-test are 32.3%, 9.3% and 24% for the CACM, CRAN and LISA, respectively. This means that the impact of the field W in the retrieval performance is not so dominant for these collections and, thus, a good representation of information from all the subfields improves performance.

Nevertheless, CISI queries have an structure of subfields which is appropriate for creating DNF formulas with several clauses. The last two columns of table 3.13 show the results for the tests with generic DNF formulas for representing queries. The best results are obtained when documents are represented by DNF formulas with a single clause and queries are represented by generic DNF formulas. In terms of average precision, this test improves 7.5% the AND-test. In Appendix A we show that the difference between the best AND-test and the best AND/OR test is statistically significant. Besides, the difference from the best AND/OR test to the BVSP test is quite significant (18.5 % in average precision).

The AND/OR-test for the LISA collection outperformed both the AND-test (7.8% in terms of average precision) and the BVSP test (36.1% in terms of average precision). The difference is even higher at low recall levels. Again, in Appendix A we show that the difference from the AND/OR-test to the AND-test is statistically significant.

The conclusions of these tests are straightforward. It seems clear that the use of more expressive formulas for representing documents is good for an IR system in terms of retrieval performance. In all the collections, the experiments that stored more expressive formulas having conjunctions and disjunctions led to significant improvements in the retrieval performance of the system.

It is also very remarkable the large difference in retrieval performance between the AND/OR tests and the BVSP tests (17.5%, 28.3%, 18.5% and 36.1% in average precision for CACM, CRAN, CISI and LISA respectively). In this sense we think that our model could become a paradigm for IR models with binary information. The use of generic DNF formulas leads to very significant improvements with respect to the classical vector space model with binary weights. The good retrieval performance results of PLBRM are very promising for applying it within realistic systems.

Regarding queries, we got some improvements when applying more expressive formulas but it is not clear whether these results can be extrapolated. We could articulate generic DNF queries

only for the CISI collection. Thus, more experiments should be done to be able to formulate an hypothesis about the impact of generic DNF queries on retrieval performance.

An important circumstance is that, since the framework is general, we can choose between simple document representations (like classical binary-weighted vectors) and more expressive document representations (generic DNF formulas) depending on the specific domain and the particular representational needs at a given moment.

Queries with several clauses

Most of the experiments presented so far utilized representations of queries having a single clause. This is because information requests in the collections have often few subfields. Queries use to have a single subfield and, then, we can solely formalize one view for each query. Only the CISI collection provides us with a query structure in which DNF formulas with several clauses make sense. We also made some other attempts to get generic DNF representation for queries. In CACM we indexed queries manually and we got improvements but this was due to the use of negations and the elimination of non-important terms. The use of logical disjunctions for queries did not improve performance even with manual indexing. Our intuition is that queries might be too small to do an appropriate separation. Furthermore, even with manual query indexing, it is well known the difficulty of translating a user interest into a boolean query. Many times, it is not easy to identify whether a conjunction or a disjunction should be used. Similar experiments were conducted in the CISI collection. Besides the usual textual format, the first 35 queries of this collection are provided as boolean expressions. Then, we translated the boolean queries into DNF queries and we accomplish experiments on these set of queries but no improvement was found with respect to our usual DNF indexing (from subfields). Nevertheless this set of queries is too small to let us to articulate a conclusive hypothesis.

Specificity

The PLBRM tests presented so far used the measure $Csim$ to compute similarity between documents and queries. $Csim$ is an approximation to the model-based measure $BRsim$ which, as argued in chapter 2, captures the criterion of exhaustivity, i.e. how much of the query is satisfied by the document. In section 2.1.3 we defined the similarity measure Csp, which is an approximation to the model-based measure $BRsp$, which captures the notion of specificity, i.e. how much of the document is satisfied by the query. In practical systems both criteria should be taken into account. In fact, two documents can mention all the topics of a query but one of them could be much more useful to the user than the other. A document mentioning only the query topics is likely more interesting for the user than a document dealing with many other topics apart from the ones mentioned by the query. Then, a measure combining the exhaustivity and the specificity criteria is desirable. Equation 3.23 defines a new similarity measure combining $Csim$ and Csp through a linear combination. The constant α is a tuning value in the interval [0,1] measuring the importance of exhaustivity vs. specificity.

$$sim(d,q) = \alpha \times Csim(d,q) + (1-\alpha) \times Csp(d,q) \tag{3.23}$$

In this section we present the results of new experiments which rank documents using this new measure instead of $Csim$. These tests aim at determining the impact of specificity on retrieval performance. The previous tests, which used the similarity measure $Csim$, can be seen as particular cases of the new tests with $\alpha = 1$.

	CACM	CRAN	CISI	LISA
Avg number of terms per document	35.86	96.27	66.26	49.48
Avg number of terms per query	12.61	9.48	30.31	32.21

Table 3.15: Average number of terms in documents and queries

Tables 3.16, 3.17, 3.18 and 3.19 depict the results obtained from the new pool of tests. In these experiments we indexed both documents and queries using only logical conjunctions. For each collection, the results of the best AND-test with $Csim$ (i.e. only exhaustivity, $\alpha = 1$) are repeated on the first column of the corresponding table. The last column of each table shows the results obtained for the best combination of exhaustivity vs specificity. We tried out values of α from 0.9 to 0 in steps of 0.1. Our interest is not to obtain an ideal value for α but to test the importance of the notion of specificity for PLBRM. The corresponding precision versus recall figures are shown in figs. 3.23, 3.24, 3.25 and 3.26.

The introduction of specificity in the measure of similarity produced homogeneous results for the four collections. Very significant improvements are obtained when the similarity measure takes into account both exhaustivity and specificity. In terms of average precision, improvements of 32.1%, 27.9%, 16.3% and 21% were achieved for the CACM, CRAN, CISI and LISA collections respectively. Regarding the best values of α, only the CRAN collection gives better results if specificity is more important than exhaustivity ($\alpha = 0.3$). Let us recall that exhaustivity is a recall-oriented measure, whereas specificity is precision-oriented. In table 3.15, we show the average number of terms in the representations of both documents and queries for the four collections. If we analyze the column of the CRAN collection, we can observe that CRAN documents are large and queries are rather small. Since documents are large, relevant and unrelevant documents likely contain many common terms. On the other hand, since queries are small, they might be not very specific and, then, they retrieve many relevant documents but many unrelevant ones as well. Then, for the CRAN collection, it is more important to improve precision than to improve recall. This leads to an optimal value of α equal to 0.3, which favors the specificity measure Csp.

We can conclude that the use of a measure combining both specificity and exhaustivity criteria benefits the system in terms of retrieval performance. In practical situations, the concrete value of α should be determined depending on the particular characteristics of the collection. Besides, more sophisticated functions combining both factors can be defined.

We also ran experiments with the new similarity measure for representations of documents and queries with conjunctions and disjunctions. However, these tests did not improve their corresponding *only-exhaustivity* tests. The problem is that the measure Csp does not capture the notion of specificity when documents are represented by generic DNF formulas (i.e. with several clauses). A document is very *specific* to a given query if it only deals with concepts mentioned by the query. Csp measures the distance from each query clause to the document as the distance from the query clause to its closest document clause. This means that only the most specific document clause is taken into account to decide how specific is the document with respect to the query. Nevertheless, a document clause can be very specific to a query but the document itself can be generic because it has other clauses dealing with many topics not mentioned by the query. Consequently, it seems obvious that, if documents have several clauses, Csp should not be used to compute the specificity of a document with respect to a query. However we can

Recall - Precision	AND-test $\alpha = 1$	AND-test $\alpha = 0.6$
0.00	0.5570	0.6268
0.10	0.4468	0.5241
0.20	0.3295	0.4514
0.30	0.2422	0.3576
0.40	0.1916	0.2795
0.50	0.1598	0.2566
0.60	0.1299	0.2024
0.70	0.0795	0.1217
0.80	0.0679	0.0922
0.90	0.0419	0.0551
1.00	0.0337	0.0438
Avg. prec.	0.2072	0.2737
% Prec. change		+32.1%
Avg. prec. for 3 intermediate points	0.1857	0.2667
% Prec. change		+43.6%

Table 3.16: CACM

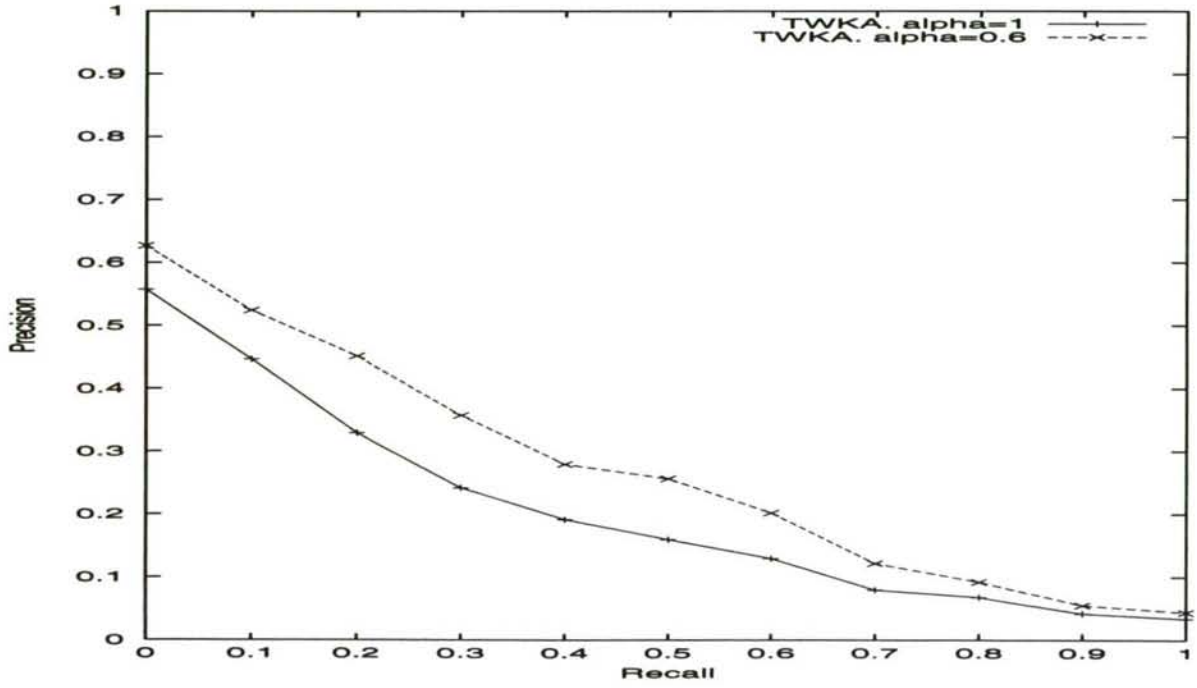

Figure 3.23: CACM - Precision vs recall graph

Recall - Precision	AND-test $\alpha = 1$	AND-test $\alpha = 0.3$
0.00	0.6946	0.8120
0.10	0.6542	0.7799
0.20	0.5340	0.6535
0.30	0.4118	0.5124
0.40	0.3291	0.4351
0.50	0.2866	0.3836
0.60	0.2131	0.2997
0.70	0.1372	0.1995
0.80	0.1025	0.1680
0.90	0.0672	0.1183
1.00	0.0654	0.1106
Avg. prec.	0.3178	0.4066
% Prec. change		+27.9%
Avg. prec. for 3 intermediate points	0.3077	0.4017
% Prec. change		+30.5%

Table 3.17: CRAN

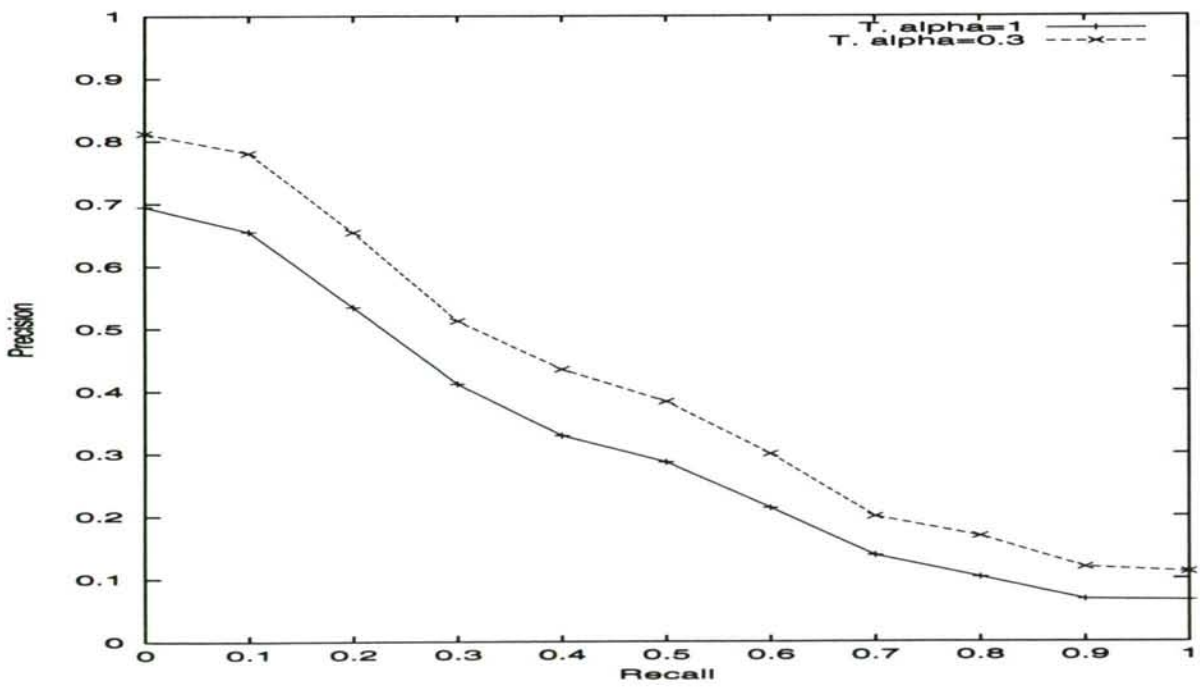

Figure 3.24: CRAN - Precision vs recall graph

Recall - Precision	AND-test $\alpha = 1$	AND-test $\alpha = 0.5$
0.00	0.4826	0.5599
0.10	0.2461	0.3240
0.20	0.1757	0.2178
0.30	0.1451	0.1712
0.40	0.1249	0.1438
0.50	0.1076	0.1266
0.60	0.0936	0.1030
0.70	0.0803	0.0857
0.80	0.0711	0.0717
0.90	0.0597	0.0556
1.00	0.0485	0.0424
Avg. prec.	0.1487	0.1729
% Prec.change		+16.3%
Avg. prec. 3 int. points	0.1181	0.1387
% Prec. change		+17.4%

Table 3.18: CISI

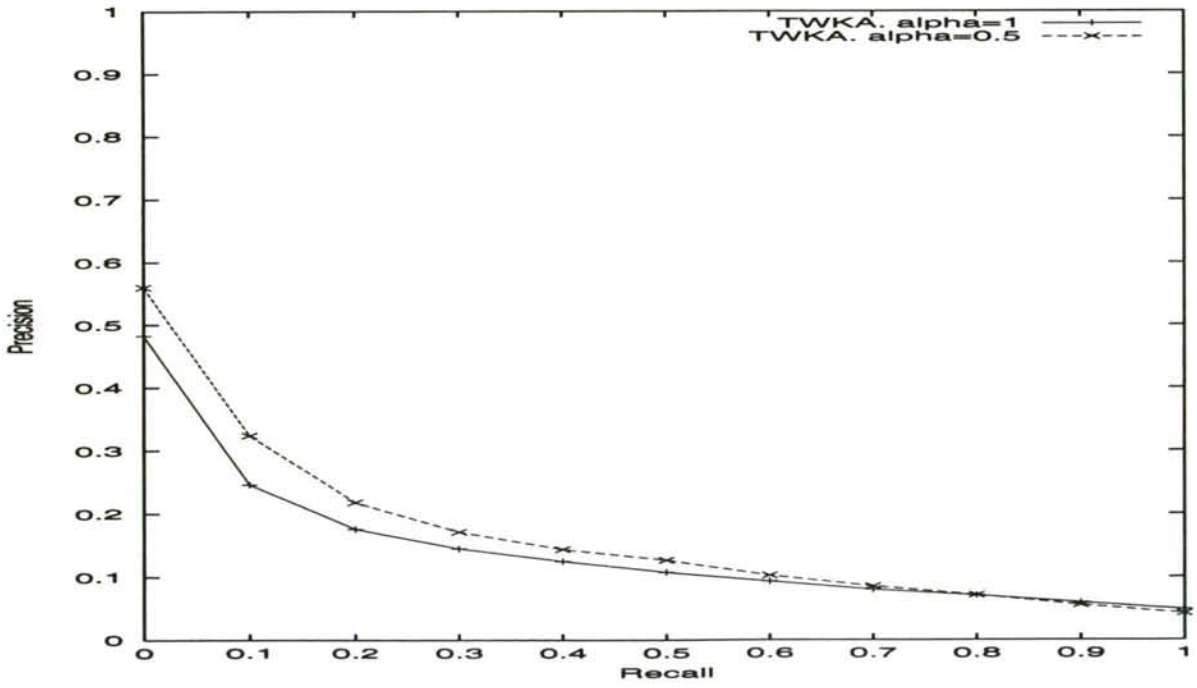

Figure 3.25: CISI - Precision vs recall graph

Recall - Precision	AND-test $\alpha = 1$	AND-test $\alpha = 0.6$
0.00	0.2095	0.2279
0.10	0.1760	0.1908
0.20	0.1295	0.1514
0.30	0.0859	0.0997
0.40	0.0256	0.0644
0.50	0.0121	0.0413
0.60	0.0062	0.0071
0.70	0.0047	0.0049
0.80	0.0041	0.0043
0.90	0.0035	0.0036
1.00	0.0031	0.0030
Avg. prec.	0.0600	0.0726
% Prec. change		+21%
Avg. prec. for 3 intermediate points	0.0485	0.0657
% Prec. change		+35.5%

Table 3.19: LISA

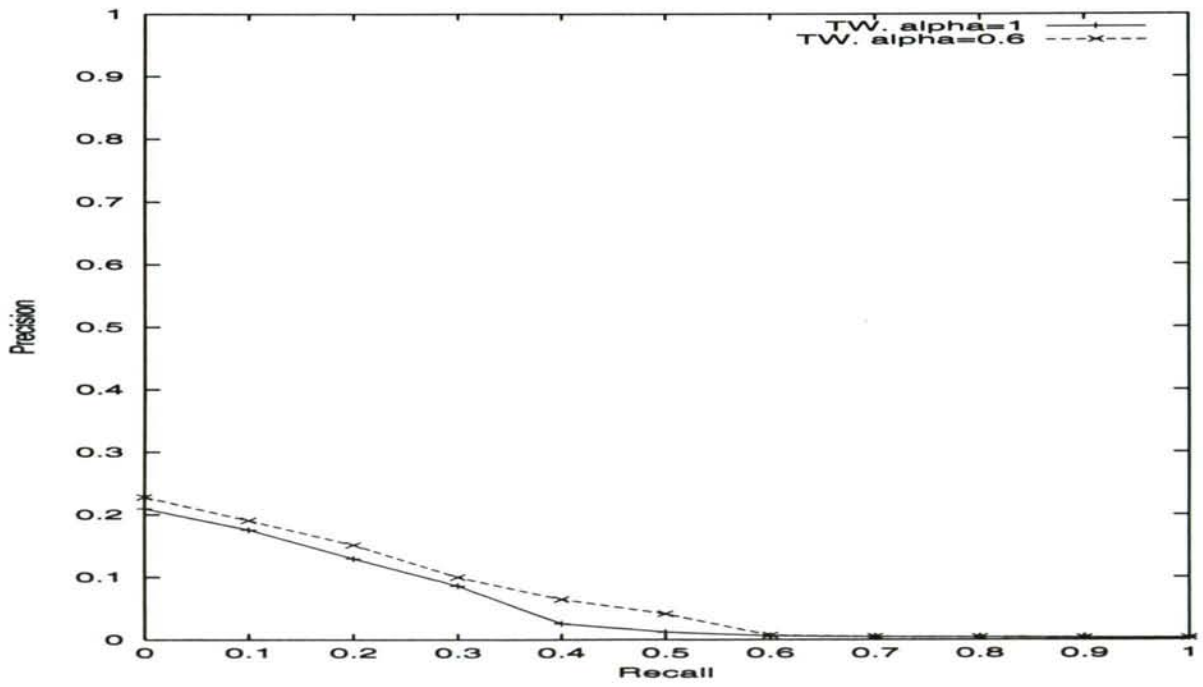

Figure 3.26: LISA - Precision vs recall graph

advance some scenarios in which the behavior of Csp for representations with several clauses can be useful. As argued above, Csp gives us the distance from each query clause to its closest document clause, i.e. to the most specific document clause. This feature could be applied for determining the part of the document which deals with the topics of the query. For instance, we could apply techniques of the area of Passage Retrieval to get a document divided into several parts (e.g. the simplest case: a document divided into its paragraphs) and each part can be modeled as a clause in a DNF formula. In this case, the measure Csp would provide us with the distance from the query to its most specific document paragraph. This sort of matching corresponds directly with some matching functions used in Passage Retrieval. Indeed, the work by Callan [9] uses the best document's passage to measure the similarity from a document to a query. The use of Csp for this kind of tasks is out of the scope of this thesis and is considered as a future line of work.

3.3.4 Conclusions

The set of experiments which we have conducted allows us to extract some interesting conclusions. First, the actual applicability of the model has been assured. Four standard test collections have been used to extract logical representations for both documents and queries, and the Algorithm A_BRsim-SC, presented in section 3.1.5, was used to rank documents in terms of their similarity to queries. A logical IR system was built and its results are comparable to the ones provided by classical IR systems. This has often been a lack in the field of logical models of IR. The model of logical imaging for IR [14] is one of the few logical models which has been tested against an standard collection. This experimentation revealed some problems derived from the size of the collection used for the evaluation [13].

The PLBRM model with documents and queries represented by logical conjunctive formulas performs roughly the same as the classical BVSP. This is not surprising because we already showed in section 2.1.3 that the inner product query-document similarity measure is subsumed by our similarity measure (for binary weights). More important results are obtained from more expressive representations of documents and queries and from the use of a similarity measure combining both exhaustivity and specificity criteria. The ability of the model for expressing several views of a document/query has been effectively used for representing distinct subfields of documents and queries. This led to improvements (very significant in some collections) in retrieval performance with respect to a flat structure of the representations (i.e. conjunctive formulas). Besides, if documents are represented by DNF formulas with a single clause, the measure Csp captures the notion of specificity of a document with respect to a query. In this case, the introduction of specificity within the similarity measure produces important improvements with respect to a ranking algorithm using only the exhaustivity criterion. Our experiments have also revealed that, if the DNF formulas have several clauses, the measure Csp is useless to capture the notion of specificity. Nevertheless, the semantics of this measure when dealing with generic DNF formulas suggests that it could be applied for other IR tasks.

An important consideration is that the experiments presented so far have not used the ability of the model to deal with negations. Propositional formulas allow us to articulate queries containing negated terms and to index documents using negated terms. Consider fig. 3.27 in which an information need from the CACM collection is shown. Clearly, the last part of the information request (namely "`We are not interested in the dynamics of arm motion.`") leads naturally to the introduction of negations in the final query representation. The vector space model cannot cope with negated terms and, thus, a representation of the request as a vector

```
.I 6
.W
 Interested in articles on robotics, motion planning particularly the
geometric and combinatorial aspects.  We are not interested in the
dynamics of arm motion.
.N
 6. John Hopcroft (robotics)
```

Figure 3.27: An information request from CACM

would not be precise. On the contrary, DNF formulas can deal with negations and, as a result, they would allow us to articulate representations which are more accurate. Then we could apply automatic techniques in order to detect situations like the one depicted in the figure and, as a consequence, we could write negations in the query. It seems sensible to think that this would increase the precision of the system in terms of retrieval performance. Indeed, a negation is a promising tool to improve precision. In the example request depicted in the figure, a negation can help to reject the documents dealing with the geometric and combinatorial aspects of motion planning in robotics that also deal with the dynamics of arm motion. These documents are not relevant to the user, as expressed in the last part of the information request. Furthermore, we ran some experiments in the CACM collection using manual query indexing. In these tests, the introduction of negations produced improvements in retrieval performance. Unfortunately, there are few CACM queries that lead naturally to the use of negations.

In general, even applying manual indexing, it is difficult to translate a plain text query into a DNF formula. Only the human who wrote the information request can determine precisely the actual semantics of his/her information need. We believe that the structure of DNF formulas can be very useful within interactive environments. The notion of a formula divided into several views is quite intuitive and we think that good user interfaces can be designed in order to guide the user when building his/her query in DNF. Besides having a split query representation, users can also take advantage from negations and, thus, articulate DNF queries which capture their needs in a better way.

Regarding documents, the indexing processes applied here have not used negations for representing documents. In systems in which documents are indexed manually, the use of negations can help the human indexer to determine precisely the contents of the document. The design of automatic procedures that extract negated terms for representing documents seems to be a great challenge. Nevertheless, statistical techniques similar to the ones applied for clustering could help in this issue.

Chapter 4

Retrieval Situations

Most IR systems decide relevance on the basis of a matching of keywords between the representations of documents and information needs. Clearly, this is a simplification because there are many other factors affecting a relevance judgment. Semantic relations between terms, particularities of the language and user's models are some examples of factors that should not be left aside at retrieval time [60, 49, 48]. For instance, the user model can include user's knowledge and his/her intentions. This information can be effectively used to improve retrieval precision. Nie and other researches [49] proposed the name of *retrieval situations* to comprise all the other factors, apart from keyword matching, affecting a relevance judgment. The following example illustrates the importance of retrieval situations. Imagine two users looking for documents on the topic "information retrieval". One of them is an IR researcher and the other is a novice student. Clearly, a document about "relevance feedback" should not be judged with the same level of relevance for both users. The novice user hardly knows that relevance feedback has something to do with IR. He/she is probably looking for more generic documents about the subject.

An adequate formalism is needed to model situational factors. For instance, introducing retrieval situations in the boolean model may lead to counterintuitive results. This is because retrieval situations can come into contradiction with the document's content, making the document to be regarded as relevant to any query. This is not a pathological case because contradictions can easily arise between situational knowledge and document's content.

Our claim is that Propositional Logic and Belief Revision are good formal tools to formalize retrieval situations. Then, in this chapter we extend the model presented in previous chapters to cope with retrieval situations. We propose to represent retrieval situations through propositional formulas. Since a retrieval situation can come into contradiction with a document representation, we propose to revise the document with the retrieval situation. The formula resulting from this revision process is used to decide relevance against queries. In this way, the relevance decision is made with more knowledge available (combining document's content and situational knowledge). Besides, we analyze the basic properties of different belief change methods and their adequacy to model the revision process between a retrieval situation and a document. One of the advantages of logical models is their generality. In this line we show that the inclusion of retrieval situations is done in a natural way. The previous model, which does not consider retrieval situations, remains as a particular case.

The rest of the chapter is organized as follows. In section 4.1 we sketch the proposal of [49], which is based on modeling retrieval situations with counterfactuals. This approach proposes a general framework where situational factors can be included but a method of implementation remains unspecified. In section 4.2 we study the connection between the evaluation of coun-

terfactuals and Belief Revision. This allows us to include retrieval situations in our model. Furthermore, in the area of Belief Revision there are several works proposing efficient implementations for the revision processes. Thus we can take advantage of these results for our application domain. Section 4.3 presents an efficient algorithm which is able to compute the revision of a retrieval situation with a document. The chapter ends with a discussion.

4.1 Counterfactuals to model retrieval situations

Formal models of IR usually consider relevance from a system point of view, isolating relevance in a restricted context in which only a matching between topics matters. This approach to IR is commonly called the *topical approach* and relevance defined in this way is also referred to as *system relevance*. It has been widely accepted that topicality is not the single criterion of a user's relevance judgment. A number of other factors also make influence on relevance judgments, e.g. the user's knowledge on the subject, the expected use of information, the particularities of the language and so on. One could accept that IR systems can achieve, at best, high levels of topicality but another alternative, which would advance the field of IR rather than admitting present limitations, would be to explore the possibility of incorporating users' relevance criteria into the retrieval mechanism itself [4]. In this sense, our aim in this chapter is to move in the second direction. Several works have studied relevance from a cognitive point of view and various terms have been proposed to encompass all these factors, such as state of the user, situationality of the user, the problem state and information need situation. Along this work we adopt the term retrieval situation to refer to all the criteria other than topicality that affect to the user's relevance judgment.

One of the reasons for the partial modeling adopted by classical models is due to inappropriateness of standard tools for describing relevance in a general manner. In this line, Nie, Brisebois and Lepage [49] identified a more appropriate logical framework for modeling relevance.

Unfortunately, classical logic is not enough to model retrieval situations. An extension to the classical logic model for IR in order to handle retrieval situations would decide relevance on the basis of the entailment $S \models (d \supset q)$, where $\supset$ is the material implication and S, d and q are the logical representations of a retrieval situation, a document and a query respectively. Nevertheless, if there is a contradiction between S and d, the previous entailment always holds for any q. Then, all queries would retrieve the document and, clearly, this is undesirable. Consequently, the classical logic model can only consider domain knowledge which is coherent with any document content in that domain. As a result, a more sophisticated formalism is needed to be able to incorporate situational factors. Nie, Brisebois and Lepage [49] proposed counterfactual conditional logic to model retrieval situations. The ability of this kind of logics to cope with contradictions was found very useful for IR.

A counterfactual implication $A > B$ is a conditional statement of the form "if A were true, B would hold", where A is assumed to be false in the current state of affairs. These implications are often named as counterfactuals or conditionals. It is exactly in the contradictory case that counterfactuals diverge from material implications. Let us illustrate the difference between a conditional implication and the material implication through the following example.

> "If the bus drivers hadn't gone on strike, I would not have been late"
>
> "If the bus drivers hadn't gone on strike, Paris would be in Spain"

According to the semantics of material implication, if the bus drivers have actually gone on

strike both sentences above are true. Nevertheless, this in counterintuitive because there is no connection between the strike and the location of Paris. Conditional statements try to capture this intuition by admitting the first sentence while rejecting the second.

Nie, Brisebois and Lepage applied the semantics of conditional implications to model users' relevance judgment. This allows to formalize the following question:

> If the information contained in the document were provided to the user in the given retrieval situation, would the query be satisfied?

In this way, relevance is captured in a more precise way. The basic model was defined as follows: given d and q, the logical representations of a document and an information need respectively, and S, a logical representation of a retrieval situation, relevance should be based on evaluating the conditional $d > q$ relative to S.

To evaluate $d > q$ relative to S we first have to revise S with respect to d such that d becomes true in the revised situation S'. Intuitively, this revision corresponds to a process after which the content of the document is accepted. The query q is then evaluated in the revised situation S'. Only if q is true in the revised situation the conditional $d > q$ is true. With this formulation $d > q$ is not true systematically when a document d contradicts the original retrieval situation S. The idea of *minimal change* is taken into account during the revision of the situation: the parts of S that are consistent with d are maintained in S' while the inconsistent ones are modified minimally in order to keep consistence.

Modeled as a counterfactual, document relevance is understood from a learning point of view:

> A document is relevant only if it contains the information such that if it is learnt, the user's information need will be satisfied.

4.2 Belief Revision to model retrieval situations

Ramsey was the pioneer on defining a test for evaluating conditional propositions [50]. Ramsey's method can be described as follows: in order to find out whether a conditional proposition is acceptable in a given state of belief, one first adds the antecedent of the conditional hypothetically to the given stock of beliefs. Second, if the antecedent together with the formerly accepted sentences leads to a contradiction, then one makes some adjustments, as small as possible without modifying the hypothetical belief in the antecedent, such that consistency is maintained. Finally, one considers whether or not the consequent of the conditional is then accepted in this adjusted state of belief.

This test has attracted a great deal of attention as a possible starting point for a formal semantics of conditionals. The Ramsey test presumes some method of *revising* states of belief. Then, it connects naturally with the Theory of Epistemic Change. In this sense, the Ramsey test was later summarized by Gärdenfors [23] within the Theory of Epistemic Change. Gärdenfors' formulation can be described as follows:

> **Ramsey test (Gärdenfors formulation):** Evaluating a counterfactual $A > B$ in a given knowledge base T is equivalent to test whether B is a logical consequence of $T \circ A$ (i.e. whether or not $T \circ A \models B$), where $T \circ A$ represents the knowledge base T revised with the formula A by the revision operator $\circ$.

4.2.1 Gärdenfors triviality result

Despite the intuitive aspect of the Ramsey test, which connects the semantics of Belief Revision and the evaluation of conditional implications, its strict application leads to the well known *Gärdenfors Triviality Result* [23]. Roughly speaking, this result claims that there are no significant logic being compatible with the Ramsey test.

Gärdenfors proved his theorem in the context of *belief sets*. A belief set is a set of formulas which are accepted in the current state of belief. It is assumed that belief sets a) include all truth-functional tautologies and b) are closed under logical consequence. Strictly speaking, belief sets are just theories in the standard logical sense. When considering computer-based knowledge bases, we need to fix a formalism and a finite syntactic representation of a knowledge base. Thus, we can assume knowledge bases represented by finite sets of propositional formulas. In this sense, we can formulate Gärdenfors triviality theorem as follows.

> **Gärdenfors Triviality Theorem:** Let **L** be a logic with a revision operator $\circ$ and the conditional connective $>$, such that **L** follows the Ramsey test and the postulates (R1), (R2) and (R3). Then **L** is trivial.

The postulates (R1), (R2) and (R3) are accepted as fundamental by any revision method. Given ψ and μ two propositional formulas, these postulates are defined as:

($R1$) $\psi \circ \mu$ implies μ

($R2$) (**Preservation Principle**) If $\psi \wedge \mu$ is satisfiable, then $\psi \circ \mu \equiv \psi \wedge \mu$

($R3$) If μ is satisfiable, then $\psi \circ \mu$ is also satisfiable

A logic is trivial if we cannot articulate four formulas, such that three of them are pairwise inconsistent and the fourth one is consistent with each of the former three. Formally, a logic L is said to be *trivial* if there are not four sentences ψ, χ, μ and ϕ in the language for L, such that the sentences $\psi \wedge \chi$, $\psi \wedge \mu$ and $\chi \wedge \mu$ are inconsistent in L, and the sentences $\phi \wedge \psi$, $\phi \wedge \chi$ and $\phi \wedge \mu$ are consistent in L. Otherwise the logic L is *non-trivial*.

The central point of Gärdenfors' result stands on the inconsistency between the *Preservation Principle*, which is a fundamental criterion in all revision methods, and the *Monotonicity Principle*, which follows directly from the Ramsey test. The former principle was formally written above as (R2) and formalizes the notion of information economy, i.e. we do not want to give up beliefs unnecessarily.

Monotonicity formalizes the notion that if a knowledge base is a consequence of another knowledge base, the same holds for their respective revisions with any formula A:

> **Monotonicity Principle:** Given K and K' two knowledge bases such that $K \models K'$, then $K \circ A \models K' \circ A$

The proof that monotonicity is a consequence of the Ramsey test can be found in [23]. The preservation condition and the Ramsey test cannot both be rational criteria for belief revisions. If both principles hold the associated logic is *trivial*.

4.2.2 Gärdenfors triviality result for IR

Our primary aim is to implement the relevance test with Belief Revision, following directly the Ramsey test. Formally, let us consider a retrieval situation modeled by a set of Propositional Logic formulas contained in a logical theory S. A document and an information need are represented by propositional formulas d and q respectively. The relevance test is implemented through a revision process as follows.

> **Relevance test:** The document d is considered relevant to the information need q in the situation S if and only if $(S \circ d) \models q$ holds.

This is a direct translation following the Ramsey test of the test based on a counterfactual as proposed by Nie, Brisebois and Lepage [49]. However, as argued before, the direct application of the Ramsey test leads to the Gärdenfors Triviality Result. As a consequence we study now whether this triviality result poses any problem for IR and, in the case, which of the solutions proposed in the literature could be appropriate.

Firstly, we analyze whether or not the triviality result is actually a problem for our application in IR. Thus, we discuss now whether a trivial logic is suitable for our purposes. Let us recall that a retrieval situation is represented by a set S of propositional formulas. A typical formula belonging to S could be a material implication formalizing a semantic relation between two terms. A document and an information need are represented by logical formulas d and q respectively. To decide relevance we have to check whether $(S \circ d) \models q$ holds. Then, the logic has to give us the ability to represent effectively documents, queries and retrieval situations. Nevertheless a trivial logic would not allow us to represent three pairwise disjoint documents that are consistent with a query. This is an unacceptable constraint, as illustrated by the following example. Let us consider three pairwise inconsistent documents represented by the formulas d_1, d_2 and d_3 respectively and a query represented by the formula q.

$$d_1 = databases \wedge \neg programming \wedge operating\ systems$$

$$d_2 = \neg databases \wedge programming \wedge operating\ systems$$

$$d_3 = databases \wedge programming \wedge operating\ systems$$

$$q = operating\ systems$$

This is a totally feasible case in IR and a trivial logic would not allow us to express the four items. As a consequence of the triviality result, a logic with the Ramsey rule and with a change operator satisfying the preservation principle is trivial. Then, this kind of logic cannot represent the above sentences and, as a result, its use is unacceptable for IR.

4.2.3 Avoiding triviality

Since Gärdenfors presented his triviality result, several researchers have studied different methods to avoid it. There have been two fundamental lines of work in this sense. Approaches on the first line drop the Preservation Principle. Although this principle is commonly accepted by Belief Revision operators, it is not satisfied by other change methods, such as Belief Update mechanisms [30]. However, the decision between update or revision should be made according to the correspondence between their semantics and the expected behavior in the domain of application. In section 4.2.4 we show that update's semantics is not suitable for our purposes. In

fact, the notion of information economy enclosed by the Preservation Principle is also desirable for IR. For the revision process $S \circ d$ (which is needed to implement the relevance test) we can instantiate the Preservation Principle as:

If $S \wedge d$ is satisfiable, then $S \circ d \equiv S \wedge d$

This means that if there is no contradiction between the retrieval situation and the document, the revised formula simply accepts both sources of knowledge. This is important for IR because if we drop beliefs unnecessarily we can damage the posterior decision of relevance with respect to the query.

Other researchers have followed the line of defining weaker forms of the Ramsey test. The key objective of this policy is to avoid monotonicity. The dependency between conditionals and change semantics is captured in a less strict way:

Weak Ramsey test: $A > B$ is *accepted* in K iff $K \circ A \models B$.

The proof that monotonicity is a consequence of the Ramsey test [23] stands on the fact that the conditional connective $>$ belongs to the language. In contrast, the previous relaxed form of the Ramsey test blocks that proof. A problem of this more relaxed formulation is that it cannot represent nested conditionals. This means that we cannot represent conditionals with the form "if (if A then B) then C". For IR this implies that we cannot represent documents using conditional sentences. However, we are not interested here in conditionals for that representational task. In fact, in [49] the counterfactual implication was proposed to make the relevance test for IR while other aspects such as relationships between terms were modeled with material implications. Our choice is to consider counterfactuals at the meta-level, to which the triviality result does not apply. Since the relevance test can be done using a revision process at the meta-level, we can follow the previous formulation of the Ramsey test such that we can decide relevance using a logic without conditionals. This goes in the line of [14], where the authors evaluate a conditional sentence without explicitly defining the conditional operator.

4.2.4 Choosing a change semantics

The key choice now is the kind of semantics that should be used to change the retrieval situation S with the document d. This section studies different change semantics which have been proposed in the literature. This is important because a change method should be selected not only for its ability to block the triviality result, but on the basis of its appropriateness for IR. Moreover, it is widely believed that there is no general-purpose, domain-independent means of updating knowledge bases that will do "the right thing" under all circumstances. Thus, this section is an attempt to determine which is the most appropriate semantics for the revision process $S \circ d$.

Belief Revision rationality postulates

The properties that a *reasonable* Belief Revision operator should have were formalized as a set of rationality postulates by Gärdenfors, Alchourrón and Makinson [1, 22, 2]. These postulates were formulated in a very general way and did not assume any concrete representation. The AGM postulates were later adapted to knowledge bases, i.e. finite sets of propositional sentences by Katsuno and Mendelzon (KM) [32]. The set of KM postulates is depicted in fig.4.1. The first three postulates were already shown above. Belief Revision rationality postulates have been

($R1$) $\psi \circ \mu$ implies μ

($R2$) If $\psi \wedge \mu$ is satisfiable, then $\psi \circ \mu \equiv \psi \wedge \mu$

($R3$) If μ is satisfiable, then $\psi \circ \mu$ is also satisfiable

($R4$) If $\psi_1 \equiv \psi_2$ and $\mu_1 \equiv \mu_2$ then $\psi_1 \circ \mu_1 \equiv \psi_2 \circ \mu_2$

($R5$) $(\psi \circ \mu) \wedge \psi$ implies $\psi \circ (\mu \wedge \psi)$

($R6$) If $(\psi \circ \mu) \wedge \psi$ is satisfiable, then $\psi \circ (\mu \wedge \psi)$ implies $(\psi \circ \mu) \wedge \psi$

Figure 4.1: Belief Revision rationality postulates for knowledge bases

widely recognized as a paradigm and it is interesting to analyze them trying to identify the implications that they impose when applied to IR. In this sense, it is important to recall that we want to check whether $(S \circ d) \models q$ holds. Thus, we revise the retrieval situation S with the document d. The following paragraphs analyze the rationality postulates in the light of the revision $S \circ d$.

The first postulate, sometimes called the *success* postulate, states that after the revision, the document should be a logical consequence of the revised situation. This is a basic postulate in Belief Revision and formalizes the notion that the knowledge base and the new information have different status because after the revision the new information has to be accepted no matter what happens with the old information. From the IR perspective, the important consideration is whether the document should prevail over the retrieval situation. Obviously, if we want to implement the counterfactual proposed in [49] we have to follow the Ramsey test whose result forces us to make the revision $S \circ d$. However one could think that the proposal could have been to test whether $S > q$ in d. In any case, we think it is interesting to discuss both options. If $S \wedge d$ is satisfiable both tests produce the same result. This follows directly from R2. When the representation of a document and the representation of a situation come into contradiction, one of the representations is changed in order to keep consistence. Let us analyze the choice through an example. A user can think that "medicine" has nothing to do with "computer science". This can be modeled as the belief `medicine` $\rightarrow \neg$ `computer science` belonging to S. An hypothetical document dealing with both issues could be represented as `medicine` $\wedge$ `computer science` $\wedge \ldots$ and would come into contradiction with S. The revision $S \circ d$ entails both medicine and computer science (from R1), so that a query about one of them would retrieve the document. This is what intuitively would be expected. On the other hand, the revision $d \circ S$ entails `medicine` $\rightarrow \neg$ `computer science` (from R1), but cannot entail both `medicine` and `computer science` (from R3). This would prevent a query containing the *missing* term from retrieving the document, even though the document actually contains that term. Therefore, an important effect of choosing $S \circ d$ is that the user can learn relationships that he/she did not know. In fact, the knowledge represented in S can be erroneous and documents can help to rectify it.

Postulate R2 says that if there is no contradiction between a document and a retrieval situation, the revision is the result of their conjunction. As mentioned above, R2 is known as the Preservation Principle and the idea of information economy captured by R2 is suitable for IR.

Postulate R3 says that although the retrieval situation were unsatisfiable, the result of the revision is satisfiable if the document is satisfiable. An unsatisfiable retrieval situation could come from a user whose knowledge is contradictory. Even in this case, a satisfiable document

($U1$) $\psi \diamond \mu$ implies μ

($U2$) If ψ implies μ then $\psi \diamond \mu \equiv \mu$

($U3$) If both ψ and μ are satisfiable then $\psi \diamond \mu$ is also satisfiable.

($U4$) If $\psi_1 \equiv \psi_2$ and $\mu_1 \equiv \mu_2$ then $\psi_1 \diamond \mu_1 \equiv \psi_2 \diamond \mu_2$

($U5$) $(\psi \diamond \mu) \wedge \psi$ implies $\psi \diamond (\mu \wedge \psi)$

($U6$) If $\psi \diamond \mu_1$ implies μ_2 and $\psi \diamond \mu_2$ implies μ_1 then $\psi \diamond \mu_1 \equiv \psi \diamond \mu_2$

($U7$) If ψ is complete then $(\psi \diamond \mu_1) \wedge (\psi \diamond \mu_2))$ implies $\psi \diamond (\mu_1 \vee \mu_2)$

($U8$) $(\psi_1 \vee \psi_2) \diamond \mu \equiv (\psi_1 \diamond \mu) \vee (\psi_2 \diamond \mu)$

Figure 4.2: Belief Update rationality postulates for knowledge bases

would keep the revision satisfiable, so that it would not entail any query. On the other hand, an unsatisfiable document d would lead to an unsatisfiable formula $S \circ d$ and, then, the document d would be considered relevant to any query. Basically, this latter case does not affect in an unreasonable way because an unsatisfiable document is a kind of somewhat *pathological* and any *normal* document will be represented by a formula that is neither valid nor unsatisfiable. An interesting discussion about this point can be found in [56].

Postulate R4 states the Principle of the Irrelevance of Syntax and the last two postulates represent the condition that revision is accomplished with minimal change [30]. To illustrate the idea behind these postulates, let us suppose that there is some metric for measuring the distance between the models of ψ, $Mod(\psi)$, and any interpretation I. Postulate (R5) says that if an interpretation I is minimal with respect to a set, $Mod(\psi)$, and I also belongs to a smaller set, $Mod(\mu \wedge \phi)$, then I must also be minimal within the smaller set $Mod(\mu \wedge \phi)$. A violation of the postulate (R6) would imply that an interpretation I may be closer to the KB than J within a certain set, while J is closer than I within some other set. The application of these postulates in the IR domain assures that a retrieval situation is changed minimally in order to accept a document.

Belief Update

In this section we analyze whether Belief Update operators are suitable for our present application. Katsuno and Mendelzon [30] showed that the original rationality postulates proposed by Alchourrón, Gärdenfors and Makinson are not adequate for every application. A fundamental distinction between revision and update was suggested. Update consists of bringing the knowledge base up to date when the world described by it changes. On the other hand, revision should be used when we are obtaining new information about a static world. Therefore, an axiomatization for Update was provided. In fig. 4.2 the set of rationality postulates for Belief Update is depicted. The symbol $\diamond$ is used to denote an operation of update.

Postulates (U1), (U4) and (U5) correspond directly to the corresponding postulates (R1), (R4) and (R5) for revision. We have already concluded that these postulates are appropriate for modeling the change between a retrieval situation and a document representation. Nevertheless, the Preservation Principle (R2), which is a basic notion for Belief Revision methods, is not fulfilled by Belief Update mechanisms. This means that even though S is consistent with d, the update of S with d, $S \diamond d$ is not guaranteed to be equivalent to $S \wedge d$. This contrasts with the

expected behavior for IR, where the final representation should reflect both sources of knowledge without dropping any one. We should only remove parts of the knowledge represented in the retrieval situation S in case of contradiction with the document representation d.

Other consideration against the use of update operators is that they do not ensure a consistent update when the original theory is not consistent. If we compare the postulate (U3) with its corresponding postulate for revision, (R3), we observe that (U3) is strictly weaker than (R3). For update, a unsatisfiable retrieval situation can lead to an unsatisfiable formula $S \diamond d$, which would retrieve any query (recall that we would use the result of $S \diamond d$ to decide relevance through the entailment $(S \diamond d) \models q$ and, if $S \diamond d$ is inconsistent, the entailment holds for any q). This situation can arise even with a satisfiable document d. Clearly, this is an unacceptable case for IR because retrieval situations can actually enclose contradictory knowledge. In this case, the retrieval situation should be ignored but, if the document representation is satisfiable, this information should effectively used to decide relevance. In fact, this is assured by the revision $S \circ d$.

To sum up, we can conclude that Belief Update operators are not a suitable tool for modeling the change that a document makes to a retrieval situation. However the semantics of the update mechanisms, which connects directly with the dynamics of the information, could be applied for other tasks. For instance, the information contained within a retrieval situation is naturally dynamic. User's knowledge changes over time and it seems reasonable to think that the representations of the retrieval situations could be updated through update operators. Nevertheless this should be analyzed in a concrete scenario.

Which Belief Revision operator for IR?

In last sections we showed that Belief Revision rationality postulates, (R1)-(R6), conform a suitable framework for modeling the change that a document makes to a retrieval situation. Since the Theory of Epistemic Change was formulated, several Belief Revision operators have been proposed. Distinct operators define distinct semantics with different behavior with respect to the rationality postulates. In this direction, a very interesting study was made by Katsuno and Mendelzon [32]. That work, besides proving the equivalence between the postulates (R1)-(R6) and the original AGM ones for belief sets, presents a review of different Belief Revision operators. It contains an analysis of the way that several operators behave with respect to the postulates. The main conclusion is that only Dalal's revision operator [17], $\circ_D$, satisfies (R1)-(R6) in a nontrivial way. Other well known operators fail to fulfill the postulates.

Since postulates capture well the intuitions behind the change a document makes to a retrieval situation, we think that Dalal's method is the best way to carry out the revision. As a consequence, the relevance test becomes:

> **Relevance test:** The document d is considered relevant to the information need q in the retrieval situation S if and only if $(S \circ_D d) \models q$ holds.

As we have already shown in chapter 2, Dalal's revision operator is model-based. When changing a retrieval situation with a document by Dalal's operator, the models of the changed situation will be those models of the document having less keywords in *disagreement* with respect to the models of the situation. Clearly, this behavior is related to IR, where there are several similarity measures counting term matches between representations. In fact, in chapter 2 we already presented the use of Dalal's operator to get a non-binary measure of the entailment $d \models q$. However, our use of Dalal's operator is different here because, now, we are interested

in the final result of the revision. If the revised situation entails the query, the document is considered relevant. In contrast, to get the similarity measure we needed the ordering among interpretations induced by Dalal's operator but not the final result of the revision process. The ordering among interpretations was used to build a ranking of documents given a query.

The new test subsumes the test $d \models q$ because when there is not situational knowledge, i.e. $S = \emptyset$, it holds that $(S \circ_D d) \equiv d$. This follows directly from the rationality postulates for revision methods. As a consequence, the tests $(S \circ d) \models q$ and $d \models q$ are equivalent if there is not situational information.

4.2.5 Partial Relevance

As shown in previous sections our proposal stands on deciding relevance using the entailment $(S \circ_D d) \models q$. However this criterion is still too strict because it cannot represent partial relevance. In this respect, we propose to use the results of the previous chapters to get a measure of the uncertainty of the new entailment. The measure of distance between interpretations formalized by Dalal's Belief Revision operator, $\circ_D$, was used to define a similarity measure between documents and queries, $BRsim$. Formally, the Belief Revision process $q \circ_D d$ builds an order induced by the query between the document's models. This order was extended to define an order between documents given a query. Therefore, the uncertainty of the entailment $d \models q$ is captured. Besides, as it has been shown, the inner product query-document similarity measure for binary-weighted vectors can be seen as a case of the similarity measure $BRsim$. Now we are able to generalize the model to consider retrieval situations: once the retrieval situation has been revised, i.e. $S \circ_D d$ is computed, a measure of the uncertainty of the entailment $(S \circ_D d) \models q$ is obtained within the Belief Revision process $q \circ_D (S \circ_D d)$. Thus we can obtain an estimation of relevance in the interval $[0, 1]$ which takes into account situational information.

4.3 Implementation

Once we have chosen the revision $S \circ_D d$ as the appropriate method to incorporate situational knowledge within our model, we explore the computational complexity of that revision process and, as a result of this analysis, efficient implementations are proposed.

The translation of model-based BR approaches into efficient algorithms has been a problem of great concern in the BR community. Several works have focused on designing translations from proposed model-based semantics into procedures suitable for implementation. In general, it is hard to analyze and identify the practical limits of Belief Revision semantics for solving large interesting problems. However while Belief Revision is intractable in theory, it is expected that certain problems can be solved within some reasonable bound. In particular, Dalal's revision is an NP-Complete problem [17] in the general case. A very interesting study about the complexity of several methods for updating and revising knowledge bases was made by Eiter and Gottlob [20]. Specifically, their work focus on the problem of given a knowledge base T, an update formula p and a formula q, decide if q is derivable from $T \circ p$. This is precisely our relevance test. An important point is that Dalal's approach gives better complexity results than the other operators. In the general case complexity is in the class $P^{NP[O(logn)]}$. However, if we assume that p, q and all the formulas belonging to T are Horn formulas and that the size of the update formula is bounded by a constant, the complexity is polynomial, $\mathcal{O}(||T|| \cdot ||q||)$, where $||T||$ and $||q||$ represent the size of the theory and the size of the query respectively. Nevertheless, the

results reached by del Val [19, 18] about Dalal's revision are even less restrictive with the form of the formulas involved.

Del Val's work [19, 18] deeply analyzes the sources of complexity of several revision and update methods and, as a result, he proposes a syntactic characterization for the logical formulas taking part in the change operation. He identified several restrictions that lead to the design polynomial-time algorithms. Particularly, the complexity results for Dalal's Belief Revision operator are very encouraging. So far we have used DNF formulas. A formula in Conjunctive Normal Form (CNF) has the form: $c_1 \wedge c_2 \wedge \ldots$ where each c_j is a disjunction of literals: $l_1 \vee l_2 \vee \ldots$. A formula in Negation Normal Form (NNF) has the form: $c_1 \wedge c_2 \wedge \ldots$ where each c_j is a disjunction of NNF formulas. In [18] del Val presented an algorithm that implements Dalal's revision in polynomial time. This algorithm takes a theory in NNF and a new information in DNF and produces the updated theory in NNF. A similar algorithm can be built for a theory in CNF, the new information in DNF and the resulting theory in CNF. In the same line, an algorithm taking a theory in DNF, a new information in DNF and producing a new theory in DNF can be designed.

At this point it is important to recall that, after the revision $S \circ_D d$ is computed, we want to apply the techniques presented in chapter 2 in order to get a non-binary measure of the entailment $(S \circ_D d) \models q$. In section 3.1 we showed that we need formulas stored in DNF in order to be able to compute efficiently the similarity measure. Namely, we need $S \circ_D d$ in DNF. This is very important because, although any propositional formula can be translated into an equivalent DNF formula, the transformation itself can take exponential time. Thus, we are interested in the algorithm that takes a theory in DNF, a new information in DNF and produces the new theory in DNF.

Del Val's work stands on a syntactic characterization that avoids any operation between models. In fact, we used this syntactic characterization to build the algorithms that compute similarity between DNF formulas. Now, we are interested in the result of the revision (i.e, the formula representing $S \circ_D d$), whereas the previous algorithms needed the values of the distances within a revision process. Next paragraphs present the syntactic equivalent of Dalal's operator that del Val proposed.

The distance between two conjunctions of literals ψ_i and μ_j was defined as the number of literals in ψ_i whose negation is in μ_j.

$$CDiff(\psi_i, \mu_j) = \{l \in \psi_i | \neg l \in \mu_j\} \tag{4.1}$$

$$CDist(\psi_i, \mu_j) = |CDiff(\psi_i, \mu_j)| \tag{4.2}$$

Given a DNF formula μ and a conjunction of literals ψ_i, the next formula collects all the conjunctions of literals $\mu_j \in \mu$ which differ minimally from ψ_i:

$$MinDiff(\psi_i, \mu) = \{\mu_j \in \mu | \forall \mu_k \in \mu, CDist(\psi_i, \mu_j) \leq CDist(\psi_i, \mu_k)\} \tag{4.3}$$

Then, the distance between a conjunction of literals ψ_i and a DNF formula μ is defined as the distance from ψ_i to its closest clauses in μ.

$$MinDist(\psi_i, \mu) = min_{\mu_j \in \mu} CDist(\psi_i, \mu_j) \tag{4.4}$$

Finally, given a formula ψ and a DNF formula μ,

$$DNF_{min}(\psi, \mu) = \{\psi_i \in DNF(\psi) | \forall \psi_j \in DNF(\psi), MinDist(\psi_i, \mu) \leq MinDist(\psi_j, \mu)\} \quad (4.5)$$

where $DNF(\psi)$ stands for the transformation of the formula ψ into DNF. If ψ is directly in DNF we can replace $DNF(\psi)$ with ψ. DNF_{min} represents the subset of clauses in $DNF(\psi)$ which fares best in terms of their distance to μ.

Given the previous formulas del Val proved the following theorem:

Theorem: *Syntactic equivalent of Dalal's revision operator*

$$\psi \circ \mu \equiv \bigvee_{\substack{\psi_i \in DNF_{min}(\psi, \mu) \\ \mu_j \in MinDiff(\psi_i, \mu)}} \bigwedge((\psi_i \setminus Diff(\psi_i, \mu_j)) \cup \mu_j)$$

From this theorem we can straightforwardly design an algorithm that computes Dalal's revision given a DNF theory and a DNF new information. The result of this revision is in DNF. The algorithm is similar to the algorithm that del Val designed for NNF theories. Figure 4.3 depicts the algorithm we have designed for DNF theories. The symbol μ_{max} stands for the size of the largest clause in μ.

The algorithm works as follows. For each clause ψ_i of the theory we compute the clauses of the new information to which ψ_i fares minimally ($MinDiff(\psi_i, \mu)$) and the distance to those closest clauses (k). The array *Distances* stores ψ_i and $MinDiff(\psi_i, \mu)$ in the index given by $k = MinDist(\psi_i, \mu)$. After analyzing all the theory clauses, the algorithm traverses the array *Distances* in ascending order until an index m has values. The index m represents the least distance from theory clauses to new information clauses. In $Distances[m]$ we have the theory clauses faring minimally to the new information (i.e. $DNF_{min}(\psi, \mu)$) and, for each $\psi_i \in DNF_{min}(\psi, \mu)$ its associated closest μ clauses (i.e. $MinDiff(\psi_i, \mu)$). The revised theory is composed by the theory clauses that fare minimally to the new information but, in these theory clauses, the literals that are in contradiction with the associated new information clause are changed to accept the new information.

Now, we analyze the complexity of the algorithm. The loop of step 3 takes $|\psi|$ iterations. The computation of $MinDiff$ in step 4 has a cost of $|\mu|\mu_{max}$. Thus, the cost of the loop from step 3 to step 6 has a worst case of $|\psi||\mu|\mu_{max}$. The cost of step 7 is $\mu_{max} + 1$. Each $\wedge((\psi_i - Diff(\psi_i, \mu_j)) \cup \mu_j)$ can be computed at a cost of $|\mu|\mu_{max}$ for each $\psi_i \in Distances[m]$ and each $\mu_j \in MinDiff(\psi_i, \mu)$. Thus, step 8 has a worst case of $|\psi||\mu|^2\mu_{max}$ steps. Putting all together we have a complexity of $\mathcal{O}(|\psi||\mu|^2\mu_{max})$. As a consequence, once we have a retrieval situation S and a document d both in DNF. We can compute the formula $S \circ_D d$ at a cost of $|S||d|^2 d_{max}$, where d_{max} is the largest clause in the document. That is, the revision $S \circ_D d$ can be made in polynomial time. As a consequence, the efficiency of the algorithm assures its efficiency when dealing with large amounts of data.

In fig. 4.4 we present an example of the execution of the algorithm for a theory ψ and a new information μ. If we apply a direct approach to compute the result of Dalal's revision for this example, we would have to compute the models of the new information (24), the models of the theory (16) and compute all the model-to-model symmetric differences. Figures 4.5, 4.6, 4.7

Algorithm Revise:

Function Revise(ψ,μ)
Input:
theory $\psi = \{\psi_1, \psi_2, \dots\}$
new information $\mu = \{\mu_1, \mu_2, \dots\}$
Output:
$\psi \circ_D \mu$

1. Mark the literals in μ with the clauses of μ in which they occur
2. Make an array Distances of size $|\mu_{max}| + 1$
   3. Extract a new clause $\psi_i \in \psi$
   4. Compute $MinDiff(\psi_i, \mu)$ and $k = MinDist(\psi_i, \mu)$
   5. Store ψ_i and $MinDiff(\psi_i, \mu)$ in Distances[k]
   6. go to step 3 until no more ψ_i's remain
7. Traverse Distances in ascending order until an index m has values

8. $$\psi \circ \mu = \bigvee_{\substack{\psi_i \in Distances[m] \\ \mu_j \in MinDiff(\psi_i, \mu)}} \bigwedge((\psi_i \setminus CDiff(\psi_i, \mu_j)) \cup \mu_j)$$
9. return($\psi \circ \mu$)

Figure 4.3: Algorithm Revise

$\mathcal{P} = \{a, b, c, d, e, f\}$

$\psi = (a \wedge b \wedge c) \vee (\neg a \wedge c \wedge e) \vee (\neg a \wedge c \wedge e \wedge \neg f)$
$\mu = (a \wedge \neg c) \vee (\neg a \wedge \neg e \wedge f)$

Algorithm Revise:

Function Revise(ψ,μ)
Input:
theory $\psi = \{\psi_1, \psi_2, \psi_3\}$, $\psi_1 = \{a, b, c\}$, $\psi_2 = \{\neg a, c, e\}$, $\psi_3 = \{\neg a, c, e, \neg f\}$
new information $\mu = \{\mu_1, \mu_2\}$, $\mu_1 = \{a, \neg c\}$, $\mu_2 = \{\neg a, \neg e, f\}$
Output:
$\psi \circ_D \mu$

1. Mark the literals in μ with the clauses of μ in which they occur
2. Make an array Distances of size 4 ($\mu_{max} = 3$)
   3. $\psi_1 = \{a, b, c\}$
   4. $CDiff(\psi_1, \mu_1) = \{c\}$, $CDist(\psi_1, \mu_1) = 1$
      $CDiff(\psi_1, \mu_2) = \{a\}$, $CDist(\psi_1, \mu_2) = 1$
      $MinDiff(\psi_1, \mu) = \{\mu_j \in \mu | \forall \mu_k \in \mu, CDist(\psi_1, \mu_j) \leq CDist(\psi_1, \mu_k)\}$
      $MinDiff(\psi_1, \mu) = \{\mu_1, \mu_2\}$
      $k = MinDist(\psi_1, \mu) = 1$
   5. Store ψ_1 and $MinDiff(\psi_1, \mu) = \{\mu_1, \mu_2\}$ in Distances[1]

   3. $\psi_2 = \{\neg a, c, e\}$
   4. $CDiff(\psi_2, \mu_1) = \{\neg a, c\}$, $CDist(\psi_2, \mu_1) = 2$
      $CDiff(\psi_2, \mu_2) = \{e\}$, $CDist(\psi_2, \mu_2) = 1$
      $MinDiff(\psi_2, \mu) = \{\mu_j \in \mu | \forall \mu_k \in \mu, CDist(\psi_2, \mu_j) \leq CDist(\psi_2, \mu_k)\}$
      $MinDiff(\psi_2, \mu) = \{\mu_2\}$
      $k = MinDist(\psi_2, \mu) = 1$
   5. Store ψ_2 and $MinDiff(\psi_2, \mu) = \{\mu_2\}$ in Distances[1]

   3. $\psi_3 = \{\neg a, c, e, \neg f\}$
   4. $CDiff(\psi_3, \mu_1) = \{\neg a, c\}$, $CDist(\psi_3, \mu_1) = 2$
      $CDiff(\psi_3, \mu_2) = \{e, \neg f\}$, $CDist(\psi_3, \mu_2) = 2$
      $MinDiff(\psi_3, \mu) = \{\mu_j \in \mu | \forall \mu_k \in \mu, CDist(\psi_3, \mu_j) \leq CDist(\psi_3, \mu_k)\}$
      $MinDiff(\psi_3, \mu) = \{\mu_1, \mu_2\}$
      $k = MinDist(\psi_2, \mu) = 2$
   5. Store ψ_3 and $MinDiff(\psi_3, \mu) = \{\mu_1, \mu_2\}$ in Distances[2]
7. Traverse Distances in ascending order until an index m has values, $m = 1$

8. $\psi \circ \mu = \bigvee_{\substack{\psi_i \in Distances[1] \\ \mu_j \in MinDiff(\psi_i, \mu)}} \bigwedge((\psi_i - CDiff(\psi_i, \mu_j)) \cup \mu_j)$
   $= (\bigwedge((\psi_1 \setminus CDiff(\psi_1, \mu_1)) \cup \mu_1)) \vee (\bigwedge((\psi_1 \setminus CDiff(\psi_1, \mu_2)) \cup \mu_2)) \vee (\bigwedge((\psi_2 \setminus CDiff(\psi_2, \mu_2)) \cup \mu_2))$
   $= (\bigwedge((\{a, b, c\} \setminus \{c\}) \cup \{a, \neg c\})) \vee (\bigwedge((\{a, b, c\} \setminus \{a\}) \cup \{\neg a, \neg e, f\})) \vee (\bigwedge((\{\neg a, c, e\} \setminus \{e\}) \cup \{\neg a, \neg e, f\}))$

   $\psi \circ \mu = (a \wedge b \wedge \neg c) \vee (\neg a \wedge b \wedge c \wedge \neg e \wedge f) \vee (\neg a \wedge c \wedge \neg e \wedge f)$

9. return($\psi \circ \mu$)

Figure 4.4: An example of the execution of the Algorithm Revise

and 4.8 present the direct computation of the revision. The first two figures show the symmetric differences between each μ model and each ψ model. The cardinality of these differences is used as a measure of distance between a model of μ and a model of ψ. The values of the model-to-model distances are shown in figures 4.7 and 4.8. Then, the distance from each model of μ to the set of models of ψ is the distance from the model of μ to its closest model in $Mod(\psi)$ (the minimum of the respective column). At this point we are able to get the set of models of the theory revised. These models are the models of μ faring minimally to ψ. It can be easily proved that the output of the Algorithm Revise and the output of the direct computation of the revision are equivalent. This is because the set of models resulting from the direct computation of Dalal's revision (depicted at the bottom of fig. 4.8) is the same than the set of models of the formula which is the output to the algorithm revise (fig. 4.4). Clearly, if the alphabet has a large number of letters, the number of models would be huge and a direct implementation of Dalal's revision cannot work. On the other hand, the run time of the algorithm revise is not affected by the size of the alphabet.

4.4 Remarks

High expressive IR systems need a model of documents and information needs closer to their actual semantics. In order to achieve that, conventional IR models have to go deeper into the modelization of situational factors. The introduction of retrieval situations in IR models can be accomplished using counterfactual conditional logic. A counterfactual $d > q$ evaluated in a retrieval situation S was proposed in the literature to decide the relevance of the document d with respect to the query q in the given retrieval situation. At this point, we can conclude that the use of a logic with a conditional connective seems to be an overshoot because the test can be done using the classic entailment and a revision operator. Furthermore, this election blocks Gärdenfors Triviality Result and, therefore, the resulting logic is non-trivial. An efficient method was also presented assuring the application of the proposed relevance test within realistic IR systems.

A fundamental aspect captured by logic is partiality. In fact, the previous model is useless with a total information assumption. Basically, retrieval situations are ignored when documents are complete theories. A document is a complete theory if for each propositional formula p either $d \models p$ or $d \models \neg p$ holds. In that case, the document has a single model and, as a result of the postulate R1 for revision, the formula $S \circ d$ has to be equivalent to d. This means that the information modeled in S is always ignored. On the other hand, when a document is a partial theory, the representation of the retrieval situation can help to complete the document's representation. Van Rijsbergen already pointed out that the most natural assumption is to consider documents as partial descriptions [60]. In fact, we can think about a realistic IR system using this policy: the document analysis phase produces the set of keywords of each document and instead of negating the rest of the system's index terms, the system maintains partial descriptions. In logic this corresponds with the fact that the system does not make a closed world assumption (CWA) but allows documents as partial theories. Given a user's information need expressed as a query and a retrieval situation containing user's knowledge, the retrieval situation helps to complete the document representation before analyzing the query. This makes a *user-oriented completion* of the document. For instance, a common user may not know that the language ml has something to do with computer science (cs) but an experienced user would probably now that they are related concepts. As a consequence, the first user's retrieval situation will not contain

$\mathcal{P} = \{a, b, c, d, e, f\}$

$\psi = (a \wedge b \wedge c) \vee (\neg a \wedge c \wedge e) \vee (\neg a \wedge c \wedge e \wedge \neg f)$
$\mu = (a \wedge \neg c) \vee (\neg a \wedge \neg e \wedge f)$

μ models $\rightarrow$ ψ models$\downarrow$	$\{a\}$	$\{a, b\}$	$\{a, d\}$	$\{a, e\}$	$\{a, f\}$	$\{a, b, d\}$
$\{a, b, c\}$	$\{b, c\}$	$\{c\}$	$\{b, c, d\}$	$\{b, c, e\}$	$\{b, c, f\}$	$\{c, d\}$
$\{a, b, c, d\}$	$\{b, c, d\}$	$\{c, d\}$	$\{b, c\}$	$\{b, c, d, e\}$	$\{b, c, d, f\}$	$\{c\}$
$\{a, b, c, e\}$	$\{b, c, e\}$	$\{c, e\}$	$\{b, c, d, e\}$	$\{b, c\}$	$\{b, c, e, f\}$	$\{c, d, e\}$
$\{a, b, c, f\}$	$\{b, c, f\}$	$\{c, f\}$	$\{b, c, d, f\}$	$\{b, c, e, f\}$	$\{b, c\}$	$\{c, d, f\}$
$\{a, b, c, d, e\}$	$\{b, c, d, e\}$	$\{c, d, e\}$	$\{b, c, e\}$	$\{b, c, d\}$	$\{b, c, d, e, f\}$	$\{c, e\}$
$\{a, b, c, d, f\}$	$\{b, c, d, f\}$	$\{c, d, f\}$	$\{b, c, f\}$	$\{b, c, d, e, f\}$	$\{b, c, d\}$	$\{c, f\}$
$\{a, b, c, e, f\}$	$\{b, c, e, f\}$	$\{c, e, f\}$	$\{b, c, d, e, f\}$	$\{b, c, f\}$	$\{b, c, e\}$	$\{c, d, e, f\}$
$\{a, b, c, d, e, f\}$	$\{b, c, d, e, f\}$	$\{c, d, e, f\}$	$\{b, c, e, f\}$	$\{b, c, d, f\}$	$\{b, c, d, e\}$	$\{c, e, f\}$
$\{c, e\}$	$\{a, c, e\}$	$\{a, b, c, e\}$	$\{a, c, d, e\}$	$\{a, c\}$	$\{a, c, e, f\}$	$\{a, b, c, d, e\}$
$\{b, c, e\}$	$\{a, b, c, e\}$	$\{a, c, e\}$	$\{a, b, c, d, e\}$	$\{a, b, c\}$	$\{a, b, c, e, f\}$	$\{a, c, d, e\}$
$\{c, d, e\}$	$\{a, c, d, e\}$	$\{a, b, c, d, e\}$	$\{a, c, e\}$	$\{a, c, d\}$	$\{a, c, d, e, f\}$	$\{a, b, c, e\}$
$\{c, e, f\}$	$\{a, c, e, f\}$	$\{a, b, c, e, f\}$	$\{a, c, d, e, f\}$	$\{a, c, f\}$	$\{a, c, e\}$	$\{a, b, c, d, e, f\}$
$\{b, c, d, e\}$	$\{a, b, c, d, e\}$	$\{a, c, d, e\}$	$\{a, b, c, e\}$	$\{a, b, c, d\}$	$\{a, b, c, d, e, f\}$	$\{a, c, e\}$
$\{b, c, e, f\}$	$\{a, b, c, e, f\}$	$\{a, c, e, f\}$	$\{a, b, c, d, e, f\}$	$\{a, b, c, f\}$	$\{a, b, c, e\}$	$\{a, c, d, e, f\}$
$\{c, d, e, f\}$	$\{a, c, d, e, f\}$	$\{a, b, c, d, e, f\}$	$\{a, c, e, f\}$	$\{a, c, d, f\}$	$\{a, c, d, e\}$	$\{a, b, c, e, f\}$
$\{b, c, d, e, f\}$	$\{a, b, c, d, e, f\}$	$\{a, c, d, e, f\}$	$\{a, b, c, e, f\}$	$\{a, b, c, d, f\}$	$\{a, b, c, d, e\}$	$\{a, c, e, f\}$

μ models $\rightarrow$ ψ models$\downarrow$	$\{a, b, e\}$	$\{a, b, f\}$	$\{a, d, e\}$	$\{a, d, f\}$	$\{a, e, f\}$	$\{a, b, d, e\}$
$\{a, b, c\}$	$\{c, e\}$	$\{c, f\}$	$\{b, c, d, e\}$	$\{b, c, d, f\}$	$\{b, c, e, f\}$	$\{c, d, e\}$
$\{a, b, c, d\}$	$\{c, d, e\}$	$\{c, d, f\}$	$\{b, c, e\}$	$\{b, c, f\}$	$\{b, c, d, e, f\}$	$\{c, e\}$
$\{a, b, c, e\}$	$\{c\}$	$\{c, e, f\}$	$\{b, c, d\}$	$\{b, c, d, e, f\}$	$\{b, c, f\}$	$\{c, d\}$
$\{a, b, c, f\}$	$\{c, e, f\}$	$\{c\}$	$\{b, c, d, e, f\}$	$\{b, c, d\}$	$\{b, c, e\}$	$\{c, d, e, f\}$
$\{a, b, c, d, e\}$	$\{c, d\}$	$\{c, d, e, f\}$	$\{b, c\}$	$\{b, c, d, e, f\}$	$\{b, c, d, f\}$	$\{c\}$
$\{a, b, c, d, f\}$	$\{c, d, e, f\}$	$\{c, d\}$	$\{b, c, e, f\}$	$\{b, c\}$	$\{b, c, d, e\}$	$\{c, e, f\}$
$\{a, b, c, e, f\}$	$\{c, f\}$	$\{c, e\}$	$\{b, c, d, f\}$	$\{b, c, d, e\}$	$\{b, c\}$	$\{c, d, f\}$
$\{a, b, c, d, e, f\}$	$\{c, d, f\}$	$\{c, d, e\}$	$\{b, c, f\}$	$\{b, c, e\}$	$\{b, c, d\}$	$\{c, f\}$
$\{c, e\}$	$\{a, b, c\}$	$\{a, b, c, e, f\}$	$\{a, c, d\}$	$\{a, c, d, e, f\}$	$\{a, c, f\}$	$\{a, b, c, d\}$
$\{b, c, e\}$	$\{a, c\}$	$\{a, c, e, f\}$	$\{a, b, c, d\}$	$\{a, b, c, d, e, f\}$	$\{a, b, c, f\}$	$\{a, c, d\}$
$\{c, d, e\}$	$\{a, b, c, d\}$	$\{a, b, c, d, e, f\}$	$\{a, c\}$	$\{a, c, e, f\}$	$\{a, c, d, f\}$	$\{a, b, c\}$
$\{c, e, f\}$	$\{a, b, c, f\}$	$\{a, b, c, e\}$	$\{a, c, d, f\}$	$\{a, c, d, e\}$	$\{a, c\}$	$\{a, b, c, d, f\}$
$\{b, c, d, e\}$	$\{a, c, d\}$	$\{a, c, d, e, f\}$	$\{a, b, c\}$	$\{a, b, c, e, f\}$	$\{a, b, c, d, f\}$	$\{a, c\}$
$\{b, c, e, f\}$	$\{a, c, f\}$	$\{a, c, e\}$	$\{a, b, c, d, f\}$	$\{a, b, c, d, e, f\}$	$\{a, b, c\}$	$\{a, c, d, f\}$
$\{c, d, e, f\}$	$\{a, b, c, d, f\}$	$\{a, b, c, d, e\}$	$\{a, c, f\}$	$\{a, c, e\}$	$\{a, c, d\}$	$\{a, b, c, f\}$
$\{b, c, d, e, f\}$	$\{a, c, d, f\}$	$\{a, c, d, e\}$	$\{a, b, c, f\}$	$\{a, b, c, e\}$	$\{a, b, c, d\}$	$\{a, c, f\}$

Figure 4.5: Model-based computation of Dalal's revision (I)

μ models → ψ models↓	$\{a,b,d,f\}$	$\{a,b,e,f\}$	$\{a,d,e,f\}$	$\{a,b,d,e,f\}$	$\{f\}$	$\{b,f\}$
$\{a,b,c\}$	$\{c,d,f\}$	$\{c,e,f\}$	$\{b,c,d,e,f\}$	$\{c,d,e,f\}$	$\{a,b,c,f\}$	$\{a,c,f\}$
$\{a,b,c,d\}$	$\{c,f\}$	$\{c,d,e,f\}$	$\{b,c,e,f\}$	$\{c,e,f\}$	$\{a,b,c,d,f\}$	$\{a,c,d,f\}$
$\{a,b,c,e\}$	$\{c,d,e,f\}$	$\{c,f\}$	$\{b,c,d,f\}$	$\{c,d,f\}$	$\{d,e,f\}$	$\{b,d,e,f\}$
$\{a,b,c,f\}$	$\{c,d\}$	$\{c,e\}$	$\{b,c,d,e\}$	$\{c,d,e\}$	$\{a,b,c\}$	$\{a,c\}$
$\{a,b,c,d,e\}$	$\{c,e,f\}$	$\{c,d,f\}$	$\{b,c,f\}$	$\{c,f\}$	$\{a,b,c,d,e,f\}$	$\{a,c,d,e,f\}$
$\{a,b,c,d,f\}$	$\{c\}$	$\{c,d,e\}$	$\{b,c,e\}$	$\{c,e\}$	$\{a,b,c,d\}$	$\{a,c,d\}$
$\{a,b,c,e,f\}$	$\{c,d,e\}$	$\{c\}$	$\{b,c,d\}$	$\{c,d\}$	$\{a,b,c,e\}$	$\{a,c,e\}$
$\{a,b,c,d,e,f\}$	$\{c,e\}$	$\{c,d\}$	$\{b,c\}$	$\{c\}$	$\{a,b,c,d,e\}$	$\{a,c,d,e\}$
$\{c,e\}$	$\{a,b,c,d,e,f\}$	$\{a,b,c,f\}$	$\{a,c,d,f\}$	$\{a,b,c,d,f\}$	$\{c,e,f\}$	$\{b,c,e,f\}$
$\{b,c,e\}$	$\{a,c,d,e,f\}$	$\{a,c,f\}$	$\{a,b,c,d,f\}$	$\{a,c,d,f\}$	$\{b,c,e,f\}$	$\{c,e,f\}$
$\{c,d,e\}$	$\{a,b,c,e,f\}$	$\{a,b,c,d,f\}$	$\{a,c,f\}$	$\{a,b,c,f\}$	$\{c,d,e,f\}$	$\{b,c,d,e,f\}$
$\{c,e,f\}$	$\{a,b,c,d,e\}$	$\{a,b,c\}$	$\{a,c,d\}$	$\{a,b,c,d\}$	$\{c,e\}$	$\{b,c,e\}$
$\{b,c,d,e\}$	$\{a,c,e,f\}$	$\{a,c,d,f\}$	$\{a,b,c,f\}$	$\{a,c,f\}$	$\{b,c,d,e,f\}$	$\{c,d,e,f\}$
$\{b,c,e,f\}$	$\{a,c,d,e\}$	$\{a,c\}$	$\{a,b,c,d\}$	$\{a,c,d\}$	$\{b,c,e\}$	$\{c,e\}$
$\{c,d,e,f\}$	$\{a,b,c,e\}$	$\{a,b,c,d\}$	$\{a,c\}$	$\{a,b,c\}$	$\{c,d,e\}$	$\{b,c,d,e\}$
$\{b,c,d,e,f\}$	$\{a,c,e\}$	$\{a,c,d\}$	$\{a,b,c\}$	$\{a,c\}$	$\{b,c,d,e\}$	$\{c,d,e\}$

μ models → ψ models↓	$\{c,f\}$	$\{d,f\}$	$\{b,c,f\}$	$\{b,d,f\}$	$\{c,d,f\}$	$\{b,c,d,f\}$
$\{a,b,c\}$	$\{a,b,f\}$	$\{a,b,c,d,f\}$	$\{a,f\}$	$\{a,c,d,f\}$	$\{a,b,d,f\}$	$\{a,d,f\}$
$\{a,b,c,d\}$	$\{a,b,d,f\}$	$\{a,b,c,f\}$	$\{a,d,f\}$	$\{a,c,f\}$	$\{a,b,f\}$	$\{a,f\}$
$\{a,b,c,e\}$	$\{c,d,e,f\}$	$\{e,f\}$	$\{b,c,d,e,f\}$	$\{b,e,f\}$	$\{c,e,f\}$	$\{b,c,e,f\}$
$\{a,b,c,f\}$	$\{a,b\}$	$\{a,b,c,d\}$	$\{a\}$	$\{a,c,d\}$	$\{a,b,d\}$	$\{a,d\}$
$\{a,b,c,d,e\}$	$\{a,b,d,e,f\}$	$\{a,b,c,e,f\}$	$\{a,d,e,f\}$	$\{a,c,e,f\}$	$\{a,b,e,f\}$	$\{a,e,f\}$
$\{a,b,c,d,f\}$	$\{a,b,d\}$	$\{a,b,c\}$	$\{a,d\}$	$\{a,c\}$	$\{a,b\}$	$\{a\}$
$\{a,b,c,e,f\}$	$\{a,b,e\}$	$\{a,b,c,d,e\}$	$\{a,e\}$	$\{a,c,d,e\}$	$\{a,b,d,e\}$	$\{a,d,e\}$
$\{a,b,c,d,e,f\}$	$\{a,b,d,e\}$	$\{a,b,c,e\}$	$\{a,d,e\}$	$\{a,c,e\}$	$\{a,b,e\}$	$\{a,e\}$
$\{c,e\}$	$\{e,f\}$	$\{c,d,e,f\}$	$\{b,e,f\}$	$\{b,c,d,e,f\}$	$\{d,e,f\}$	$\{b,d,e,f\}$
$\{b,c,e\}$	$\{b,e,f\}$	$\{b,c,d,e,f\}$	$\{e,f\}$	$\{c,d,e,f\}$	$\{b,d,e,f\}$	$\{d,e,f\}$
$\{c,d,e\}$	$\{d,e,f\}$	$\{c,e,f\}$	$\{b,d,e,f\}$	$\{b,c,e,f\}$	$\{e,f\}$	$\{b,e,f\}$
$\{c,e,f\}$	$\{e\}$	$\{c,d,e\}$	$\{b,e\}$	$\{b,c,d,e\}$	$\{d,e\}$	$\{b,d,e\}$
$\{b,c,d,e\}$	$\{b,d,e,f\}$	$\{b,c,e,f\}$	$\{d,e,f\}$	$\{c,e,f\}$	$\{b,e,f\}$	$\{e,f\}$
$\{b,c,e,f\}$	$\{b,e\}$	$\{b,c,d,e\}$	$\{e\}$	$\{c,d,e\}$	$\{b,d,e\}$	$\{d,e\}$
$\{c,d,e,f\}$	$\{d,e\}$	$\{c,e\}$	$\{b,d,e\}$	$\{b,c,e\}$	$\{e\}$	$\{b,e\}$
$\{b,c,d,e,f\}$	$\{b,d,e\}$	$\{b,c,e\}$	$\{d,e\}$	$\{c,e\}$	$\{b,e\}$	$\{e\}$

Figure 4.6: Model-based computation of Dalal's revision (II)

μ models $\rightarrow$ ψ models$\downarrow$	$\{a\}$	$\{a,b\}$	$\{a,d\}$	$\{a,e\}$	$\{a,f\}$	$\{a,b,d\}$
$\{a,b,c\}$	2	1	3	3	3	2
$\{a,b,c,d\}$	3	2	2	4	4	1
$\{a,b,c,e\}$	3	2	4	2	4	3
$\{a,b,c,f\}$	3	2	4	4	2	3
$\{a,b,c,d,e\}$	4	3	3	3	5	2
$\{a,b,c,d,f\}$	4	3	3	5	3	2
$\{a,b,c,e,f\}$	4	3	5	3	3	4
$\{a,b,c,d,e,f\}$	5	4	4	4	4	3
$\{c,e\}$	3	4	4	2	4	5
$\{b,c,e\}$	4	3	5	3	5	4
$\{c,d,e\}$	4	5	3	3	5	4
$\{c,e,f\}$	4	5	5	3	3	6
$\{b,c,d,e\}$	5	4	4	4	6	3
$\{b,c,e,f\}$	5	4	6	4	4	5
$\{c,d,e,f\}$	5	6	4	4	4	5
$\{b,c,d,e,f\}$	6	5	5	5	5	4
$Dist(Mod(\psi),m)$	2	**1**	2	2	2	**1**

μ models $\rightarrow$ ψ models$\downarrow$	$\{a,b,e\}$	$\{a,b,f\}$	$\{a,d,e\}$	$\{a,d,f\}$	$\{a,e,f\}$	$\{a,b,d,e\}$
$\{a,b,c\}$	2	2	4	4	4	3
$\{a,b,c,d\}$	3	3	3	3	5	2
$\{a,b,c,e\}$	1	3	3	5	3	2
$\{a,b,c,f\}$	3	1	5	3	3	4
$\{a,b,c,d,e\}$	2	4	2	5	4	1
$\{a,b,c,d,f\}$	4	2	4	2	4	3
$\{a,b,c,e,f\}$	2	2	4	4	2	3
$\{a,b,c,d,e,f\}$	3	3	3	3	3	2
$\{c,e\}$	3	5	3	5	3	4
$\{b,c,e\}$	2	4	4	6	4	3
$\{c,d,e\}$	4	6	2	4	4	3
$\{c,e,f\}$	4	4	4	4	2	5
$\{b,c,d,e\}$	3	5	3	5	5	2
$\{b,c,e,f\}$	3	3	5	6	3	4
$\{c,d,e,f\}$	5	5	3	3	3	4
$\{b,c,d,e,f\}$	4	4	4	4	4	3
$Dist(Mod(\psi),m)$	**1**	**1**	2	2	2	**1**

Figure 4.7: Model-based computation of Dalal's revision (III)

μ models $\rightarrow$ ψ models$\downarrow$	$\{a,b,d,f\}$	$\{a,b,e,f\}$	$\{a,d,e,f\}$	$\{a,b,d,e,f\}$	$\{f\}$	$\{b,f\}$
$\{a,b,c\}$	3	3	5	4	4	3
$\{a,b,c,d\}$	2	4	4	3	5	4
$\{a,b,c,e\}$	4	2	4	3	3	4
$\{a,b,c,f\}$	2	2	4	3	3	2
$\{a,b,c,d,e\}$	3	3	3	2	6	5
$\{a,b,c,d,f\}$	1	3	3	2	4	3
$\{a,b,c,e,f\}$	3	1	3	2	4	3
$\{a,b,c,d,e,f\}$	2	2	2	1	5	4
$\{c,e\}$	6	4	4	5	3	4
$\{b,c,e\}$	5	3	5	4	4	3
$\{c,d,e\}$	5	5	3	4	4	5
$\{c,e,f\}$	5	3	3	4	2	3
$\{b,c,d,e\}$	4	4	4	3	5	4
$\{b,c,e,f\}$	4	2	4	3	3	2
$\{c,d,e,f\}$	4	4	2	3	3	4
$\{b,c,d,e,f\}$	3	3	3	2	4	3
$Dist(Mod(\psi),m)$	**1**	**1**	2	**1**	2	2

μ models $\rightarrow$ ψ models$\downarrow$	$\{c,f\}$	$\{d,f\}$	$\{b,c,f\}$	$\{b,d,f\}$	$\{c,d,f\}$	$\{b,c,d,f\}$
$\{a,b,c\}$	3	5	2	4	4	3
$\{a,b,c,d\}$	4	4	3	3	3	2
$\{a,b,c,e\}$	4	2	5	3	3	4
$\{a,b,c,f\}$	2	4	1	3	3	2
$\{a,b,c,d,e\}$	5	5	4	4	4	3
$\{a,b,c,d,f\}$	3	3	2	2	2	1
$\{a,b,c,e,f\}$	3	5	2	4	4	3
$\{a,b,c,d,e,f\}$	4	4	3	3	3	2
$\{c,e\}$	2	4	3	5	3	4
$\{b,c,e\}$	3	5	2	4	4	3
$\{c,d,e\}$	3	3	4	4	2	3
$\{c,e,f\}$	1	3	2	4	2	3
$\{b,c,d,e\}$	4	4	3	3	3	2
$\{b,c,e,f\}$	2	4	1	3	3	2
$\{c,d,e,f\}$	2	2	3	3	1	2
$\{b,c,d,e,f\}$	3	3	2	2	2	1
$Dist(Mod(\psi),m)$	**1**	2	**1**	2	**1**	**1**

$$Mod(\psi \circ_D \mu) = Min(Mod(\mu), \leq_\psi) = \{\{a,b\},\{a,b,d\},\{a,b,e\},\{a,b,f\},\{a,b,d,e\},\{a,b,d,f\},\{a,b,e,f\},\{a,b,d,e,f\},\{c,f\},\{b,c,f\},\{c,d,f\},\{b,c,d,f\}\}$$

Figure 4.8: Model-based computation of Dalal's revision (IV)

anything about that relation and the retrieval situation of the experienced user will contain $ml \rightarrow cs$. Then, given a document dealing with ml, if both users articulate a query asking about cs documents, the first user would not access the document and the second one would. This goes in the line that unexperienced users receive generic documents while experienced users receive specific ones. Indeed, it does not have much sense to present a ml document to a user that does not know what it is. He/she is probably looking for more general documents about computer science. Then, the precision of the set of retrieved documents is improved, saving the user from inspecting documents that, almost certainly, will not interest him.

If we want to apply the algorithm revise to implement the revision process between a retrieval situation and a document we need to store the representation of both the retrieval situation and the document in DNF. Regarding documents, DNF formulas can represent classical vectors and we already advanced in previous chapters how a DNF formula can be used to express several views of a document. In this sense we just have to articulate an indexing process that builds DNF representations for documents. With respect to retrieval situations we have to investigate methods to get DNF formulas. Resources like thesauri have tendency to CNF representations. Relations like $algebra \supset maths$ and $pascal \supset programming$ would produce representations like $(algebra \supset maths) \wedge (pascal \supset programming)$, which is equivalent to the CNF formula $(\neg algebra \vee maths) \wedge (\neg pascal \vee programming)$. As discussed above, a CNF formula can be translated into a DNF formula but taking exponential time. Thus, we could model very few relationships within a retrieval situation. However the algorithm can be efficiently applied when retrieval situations are naturally expressed in DNF. User profiles can be an example of this case because different interests can be stored in separate conjunctions. In fact, in [36] user profiles in Information Filtering are updated using Belief Revision techniques.

Some IR models have tried to estimate the uncertainty of the implication by the conditional probability $P(q/d)$. However the limitation of these approaches was shown by Lewis' triviality results [38]. Basically, only four values of probability are obtained. An important point is that Lewis' triviality is a particular case of Gärdenfors triviality [23]. Crestani and Van Rijsbergen developed a model [14] which evaluates the uncertainty of the conditional implication based on an *imaging* process. The model exploits term-term relationships and it is free from triviality. However, it does not consider retrieval situations. Nie and other researchers [49] took into account retrieval situations but their proposal is a general framework within which IR may be considered. In some sense we have described a way of implementing their proposal and the complexity of the problems involved was analyzed.

Chapter 5

Relevance Feedback

Most users of IR systems find it difficult to formulate good queries. Often there is a significant gap between user's queries and user's needs. Clearly, to provide increasingly better ranked results based solely on the initial query is limited. This suggests that the first query formulation should be treated as a naive attempt to retrieve relevant information. As a result, IR systems use to articulate methods that change the initial user's query. The aim is to obtain a reformulation of the query that reflects user's intentions in a better way.

There is a variety of approaches for improving user's initial query through *query expansion* and *term reweighting*. Query expansion consists on adding new terms to the original query and term reweighting is based on modifying weights of terms existing in the original query. Depending on the source of information used to modify the original query, approaches for query modification are grouped in three main categories:

- Approaches based on feedback from the user.
- Approaches based on information derived from the set of documents retrieved by the initial query (automatic local analysis).
- Approaches based on global information derived from the whole document collection (automatic global analysis).

The first approach requires interaction with the user whereas the latter approaches build automatically a new query. Local strategies examine the set of documents retrieved by the initial query to determine terms for query expansion. There is a variety of techniques for selecting terms. Local clustering and local context analysis are two of the most popular methods applied in automatic local analysis. Basically, clustering techniques build global structures such as association matrices which quantify term correlations (e.g. the number of documents in which two given terms co-occur). The correlated terms are used for query expansion. Local context analysis techniques use noun groups (e.g. two adjacent nouns) instead of simple keywords as document's concepts. For query expansion, concepts are selected from the top ranked documents based on their co-occurrence with query terms. However, instead of documents, passages (i.e. a text window of fixed size) are used for determining co-occurrence.

Automatic global analysis methods expand the query using information from the whole set of documents in the collection. Usually thesaurus-like structures are built using all the documents in the collection. Distinct approaches apply distinct techniques for building the thesaurus. Besides, there are different procedures for selecting terms for query expansion. Approaches based on

a similarity thesaurus build a thesaurus based on term to term relationships rather than on a matrix of co-occurrence. Term to term relationships are not derived directly from co-occurrence of terms inside documents but they are obtained by considering that terms are concepts in a concept space. Once the thesaurus has been built, terms for expansion are selected based on their similarity to the whole query rather than on their similarities to individual query terms. An alternative approach is to build an statistical thesaurus, which is composed of classes which group correlated terms in the context of the whole collection. Such correlated terms are then used to expand the original user query. To be effective, terms selected for expansion must have high term discrimination values which implies that they must be low frequency terms.

Relevance feedback presents the following advantages over other query reformulation strategies, namely:

- it shields the users from the details of the reformulation process because all the user has to provide is a relevance judgment.
- it breaks down the whole searching task into a sequence of small steps which are easier to understand.
- it provides a controlled process designed to emphasize the important information and de-emphasize the non-important information.

Although relevance feedback increases the complexity of the user interfaces, many works have focused on designing appropriate interfaces which help users to grasp the feedback process.

In this chapter we discuss how our logical model can deal with user relevance feedback. Classical models are rather rigid when handling feedback information. Query expansion cannot be handled by all classical models and negative feedback is often either disregarded or sparingly used. In this sense, we will stress the advantages of the logical model when changing the query through user relevance feedback.

5.1 User relevance feedback

Relevance feedback is the most popular query reformulation strategy. The basic idea is that relevant documents resemble each other and, if we move the original query towards *some* relevant documents the performance of the system will be improved. In order to get some relevant documents, interaction with the user is needed. The basic relevance cycle works as follows. A first retrieval is done with the user's original query and the user is presented with a list of the top-ranked documents. The user marks those documents which are relevant and the system uses this information to build a new query. The new query is expected to be a better representation of the user's information need.

User relevance feedback is widely recognized as an effective mechanism to improve performance. Early experiments using the SMART system and experiments using the probabilistic model showed very significant improvements in performance using relevance feedback for small test collections. For many years researchers have suggested different feedback strategies having much better performance than no-feedback retrieval. Original relevance feedback works used information from all the terms in the retrieved documents for changing the query. More recent proposals define techniques to get selected terms from the retrieved documents and only the selected terms are used to modify the query.

5.1.1 A logical view of feedback

In this section we present the formal details of relevance feedback. We use the vector space model to illustrate the basic notions underlying the feedback process. Besides, the formalization of the process of relevance feedback within the logical model is depicted.

The ideal case

Let us consider the ideal case in which the whole set of relevant documents to a given query is known in advance. In such a situation, it can be demonstrated that the best query vector is:

$$\vec{q_{opt}} = \frac{1}{|C_r|} \sum_{\forall \vec{d_j} \in C_r} \vec{d_j} - \frac{1}{N - |C_r|} \sum_{\forall \vec{d_j} \notin C_r} \vec{d_j} \tag{5.1}$$

where C_r is the set of relevant documents among all documents in the collection and N is the size of the document collection. The basic idea behind this formula is that the query gets closer to the terms of the relevant documents and it moves away from the terms of the non-relevant documents.

Let us translate these ideas into the framework provided by a logical model. Let q be the logical representation of a query and let $C_r = \{d_{r1}, \ldots, d_{rk}\}$ be the set of relevant documents to q, where each d_{ri} is the logical representation of a document. If we consider a basic logical model that decides relevance using the logical entailment, the optimal query q_{opt} has to retrieve all the relevant documents and it should not retrieve any non-relevant document. Formally:

- $\forall_{d_{ri} \in C_r} d_{ri} \models q_{opt}$
- $\forall_{d_{nri} \notin C_r} d_{nri} \not\models q_{opt}$

An optimal query satisfying the first condition is $q_{opt.a} = d_{r1} \vee \cdots \vee d_{rk}$. It holds that $\forall_{d_{ri} \in C_r} d_{ri} \models q_{opt.a}$ because $Mod(q_{opt.a}) = Mod(d_{r1}) \cup \cdots \cup Mod(d_{rk})$ and, then, for any i $Mod(d_{ri}) \subseteq Mod(q_{opt.a})$.

On the other hand, if we want the second condition to be satisfied we can articulate an optimal query with the form $q_{opt.b} = \neg d_{nr1} \wedge \cdots \wedge \neg d_{nrn}$, where each d_{nri} is a non-relevant document, i.e. it does not belong to C_r. The set of models of $q_{opt.b}$ does not contain any model of any d_{nri} and, thus, the second condition holds. However, the formula $q_{opt.b}$ can be unsatisfiable. For instance, given the two non-relevant documents $d_{nr1} = a \vee \neg b$ and $d_{nr2} = a \vee b$, the formula $\neg d_{nr1} \wedge \neg d_{nr2}$ is not satisfiable (applying Morgan's law it can be rewritten as $\neg a \wedge b \wedge \neg a \wedge \neg b$). Unsatisfiable queries should be avoided because given an unsatisfiable query q, the entailment $d \models q$ does not hold whichever the document d is. As a result, an unsatisfiable query would not retrieve any document, relevant or not. Clearly, this is unacceptable. In order to get a satisfiable formula that fulfills the second condition depicted above, we can take advantage from the notion of revision. We want several formulas, i.e. $\neg d_{nr1}, \ldots, \neg d_{nrn}$, to be fulfilled but their conjunction can be unsatisfiable. If we apply a revision process involving the previous formulas we can assure that, if the conjunction of them is satisfiable, the result of the revision is equivalent to the conjunction of the formulas. This behavior is obtained directly from the semantics of revision methods. Specifically, given two formulas a and b and a revision operator $\circ$, if $a \wedge b$ is satisfiable then the revision $a \circ b$ is equivalent to $a \wedge b$. Furthermore, if the conjunction is not satisfiable the revision process builds a satisfiable formula (as long as b is satisfiable) which drops as few old beliefs as

possible. This means that we can build a satisfiable formula that comes as close as possible to the desired behavior expressed by the second condition. It is important to recall that after the revision the new information has to be a logical consequence of the revised formula, i.e. $(a \circ b) \models b$. Within the revision process, if we have to drop beliefs we will drop beliefs from a. This implies that the new information is preferred with regard to the old information. Nevertheless in the ideal case the set of non-relevant documents is not ordered, i.e. all the non-relevant documents are equally unrelevant. Then, any of the possible orders of the revision processes is equally sensible. For instance, we can propose to use a formula such as $q_{opt.b'} = \neg d_{nr1} \circ \cdots \circ \neg d_{nrn}$, where we assume left associativity of the operator $\circ$. As argued before, $q_{opt.b'} \equiv q_{opt.b}$ if $q_{opt.b}$ is satisfiable. If $q_{opt.b}$ is unsatisfiable, $q_{opt.b'}$ is satisfiable but it maintains as information from the original $q_{opt.b}$ as possible.

One could argue that $q_{opt.a}$ can also be unsatisfiable. However this case only arises when all the relevant documents are unsatisfiable and that situation is not feasible in realistic systems.

Once we have an optimal query to retrieve relevant documents, $q_{opt.a}$, and an optimal query to reject non-relevant documents, $q_{opt.b}$, we have to articulate a method to build a single query. Again, a direct approach, i.e. $q_{opt.a} \wedge q_{opt.b}$, can be unsatisfiable. We can apply the same intuitions used before and, hence, we can benefit from the advantages provided by the semantics of revision. We can maintain as much information as possible from $q_{opt.a}$ and $q_{opt.b}$ through a revision process. Classical feedback use to give more importance to positive feedback (information from relevant documents) with regard to negative feedback (information from unrelevant documents). Then, a satisfiable formula can be obtained through the revision $q_{opt.b} \circ q_{opt.a}$.

The real case

In a realistic situation the set of relevant documents is not known. In the vector space model, a first retrieval is done with the initial query vector and the user analyzes the first 10 or 20 top ranked documents marking those documents which are relevant. The initial query vector is incrementally changed using the information supplied by the user. The incremental change follows the policy presented for the optimal case but the relevant documents considered are the top-ranked documents marked by the user as relevant. The next formula presents Rocchio's classical method [51] to calculate the modified query vector.

$$\vec{q_m} = \alpha \vec{q} + \frac{\beta}{|D_r|} \sum_{\forall \vec{d_j} \in D_r} \vec{d_j} - \frac{\gamma}{|D_n|} \sum_{\forall \vec{d_j} \in D_n} \vec{d_j} \tag{5.2}$$

where D_r is the subset of the retrieved documents identified as relevant by the user, D_n is the set of non-relevant documents among the retrieved documents and α, β and γ are tuning constants. It has been largely demonstrated that the following inequation should hold: $\beta > \gamma$. This means that the information from relevant documents is more important that the information from unrelevant documents. The relative importance of the original query with respect to the feedback information is strongly determined by the quality of the original query. Nevertheless, in general, it is believed that the feedback information should prevail over the information from the original query [8], i.e. it should hold that $\beta > \gamma > \alpha$.

Distinct formulations exist in the literature to change the original query vector. Our intention is to grasp the intuitive meaning of the feedback process in order to make an appropriate translation into logic and not to make a complete overview of the work in the area. There are

three actors in the previous formula: the original query, the information from relevant documents and the information from unrelevant documents. Observe that the original query is not left aside. This is because the original query may contain important information for determining user's interests. Perhaps the relevant documents in the top N resemble each other and, thus, they are on their own capable to retrieve some other relevant documents which are similar to them. Nevertheless, there can be other relevant documents which are not similar to the relevant documents in the top N but they are closer to the original query.

Let us consider the feedback process within the logical model. An initial logical query q retrieves a set of documents. The set $D_r = \{d_{r1}, \ldots, d_{ri}\}$ contains the relevant documents in the top N and the set $D_n = \{d_{nr1}, \ldots, d_{nrj}\}$ contains the non-relevant documents in the top N. We propose to apply the following revision process in order to obtain a new query:

$$q_m = q \circ ((\neg d_{nrj} \circ \neg d_{nrj-1} \circ \cdots \circ \neg d_{nr1}) \circ (d_{r1} \vee \cdots \vee d_{ri})) \tag{5.3}$$

The new query gets closer to the retrieved relevant documents and it moves away from the retrieved non-relevant documents. The importance of the factors involved determined the order of the revision processes. The formula $(\neg d_{nrj} \circ \neg d_{nrj-1} \circ \cdots \circ \neg d_{nr1})$ favors the unrelevant documents that are in a higher position in the rank (d_{nr1} is supposed to be the highest ranked non-relevant document and d_{nrj} the lowest ranked non-relevant document). In fact, some classical approaches apply negative feedback using information coming only from the highest ranked non-relevant document. To illustrate this, the following equation presents the Ide-Dec-Hi approach [29] to calculate the modified query vector in the vector space model. This approach only considers negative information from $max_{non-relevant}\ (\vec{d_j})$, which is a reference to the highest ranked non-relevant document.

$$\vec{q_m} = \alpha\vec{q} + \beta \sum_{\forall \vec{d_j} \in D_r} \vec{d_j} - \gamma\ \ max_{non-relevant}\ (\vec{d_j}) \tag{5.4}$$

The formula $((\neg d_{nrj} \circ \neg d_{nrj-1} \circ \cdots \circ \neg d_{nr1}) \circ (d_{r1} \vee \cdots \vee d_{ri}))$ makes that positive feedback prevails over negative feedback as in classical approaches ($\beta > \gamma$). The final revision process favors the feedback information with respect to the original query ($\beta, \gamma > \alpha$). Consequently, the formula 5.3 maintains the behavior of the classical approaches regarding the relative importance of old query, positive and negative feedback.

5.1.2 Implementation

The logical model for IR presented in previous chapters is efficiently applied if documents and queries are represented as DNF formulas. In order to include relevance feedback in the model, we have to assure that, given an initial DNF query, the relevance feedback process builds a new query which is in DNF. Formally, in the equation 5.3, q_m has to be in DNF if q is in DNF.

Document oriented feedback

In section 4.3 we presented a polynomial-time algorithm, *Revise*, that builds the result of Dalal's revision as a DNF formula. A DNF theory and a DNF new information are the inputs to this algorithm. User queries can be translated into DNF and, as presented in section 3.3.3, queries from standard IR collections can be represented by DNF formulas. Then, the original query q

can easily be in DNF. The new information of the formula 5.3 is, again, the result of a revision. The formula $(d_{r1} \vee \cdots \vee d_{ri})$ is directly in DNF as long as every d_{ri} is in DNF. This is because a disjunction of DNF formulas is a DNF formula. However, each formula $\neg d_{urj}$ is in CNF (a negation of a DNF formula is a CNF formula), and thus, the formula $(\neg d_{urj} \circ \cdots \circ \neg d_{ur1})$ is a revision of several CNF formulas. Unfortunately, no efficient algorithms have been designed to implement this kind of revision. Then, an efficient implementation of the feedback process should avoid the information from unrelevant documents. As a result, we consider the feedback process as a positive relevant feedback process (only information from relevant documents is taken into account):

$$q_m = q \circ (d_{r1} \vee \cdots \vee d_{ri}) \tag{5.5}$$

Now, q_m can be computed by the algorithm *Revise* because, as it has been argued before, both q and $(d_{r1} \vee \cdots \vee d_{ri})$ are in DNF. In section 5.2 we present the evaluation of this relevance feedback approach against some standard IR collections.

In the logical model of feedback presented here, unrelevant documents have to be incorporated to the query as negated documents, i.e. $\neg d_{urj}$. The structure of documents as DNF formulas makes that the formulas $\neg d_{urj}$ are in CNF and, thus, efficient algorithms cannot be applied. However, negative feedback could be used provided that the revising formulas are simpler. Instead of using directly the representations of documents for modifying the query, we can build simpler negative formulas that update the original query. In this line, we propose now a variation of the relevance feedback process where both positive and negative information is considered.

Term oriented feedback

The model of feedback presented before uses all the terms of every document for expanding the query. This is because we use the whole representation of every retrieved document for revising the original query. Some works about relevance feedback [25] have shown that expanding the original query with well-selected terms produces better performance results than expanding with all terms of the retrieved documents. We propose to apply classical techniques for selecting good terms in the retrieved documents and the resulting set of terms is used to build the formula that revises the original query. Nevertheless classical approaches only select terms on the relevant documents. This is because classical models do not have adequate tools to introduce terms from non-relevant documents. We propose to apply the same classical techniques to select good terms that characterize unrelevant documents. Since we do not want to retrieve non-relevant documents, these representative terms should be included in the query as negated terms. This is possible because the logical framework gives us the ability to write negations. This is not the case in most of the classical models. Furthermore, the introduction of negated terms is expected to produce improvements in retrieval performance because non-relevant documents can be moved down in the ranking. In summary, the main benefits of applying term selection within our model are:

- negative information can be incorporated to the query modification process.
- better evaluation results are expected.

We propose to revise the query using the selected terms from the relevant documents as positive terms and the selected terms from the unrelevant documents as negative terms. We

build a revising formula which is the conjunction of all the terms, either positive or negative. Equation 5.6 shows the new revision process.

$$q_m = q \circ (t_{p1} \wedge \cdots \wedge t_{pn} \wedge \neg t_{n1} \wedge \cdots \wedge \neg t_{nm}), \tag{5.6}$$

where $t_{p1}, \ldots, t_{pn}$ are the selected terms from the retrieved relevant documents and $t_{n1}, \ldots, t_{nm}$ are the selected terms from the retrieved non-relevant documents. Note that the number of positive and negative terms does not have to be the same.

Term selection

Several methods have been proposed to select terms for query expansion. Basically, all the terms from retrieved relevant documents are collected and ordered by a given sorting technique. Harman [24] showed that choosing the top 20 terms from the sorted list produces significant improvements in performance. The selected terms are supposedly the important ones within the set of relevant documents and, thus, they are a better representation for the set of relevant documents. As argued before, we apply the same techniques for sorting the list of terms from non-relevant documents.

Several statistical techniques have been defined for sorting the list of terms. However not all of them are applicable within our model. The term selection process has to be adapted to our representation of documents. Some classical methods use measures of frequencies of terms within documents. Propositional DNF formulas are able to represent whether or not a term appears within a document but non-binary frequencies cannot be handled. Then, this kind of statistical approaches cannot be applied within our model. We have adopted the *postings* method. The posting of a term is the number of relevant documents within the retrieved documents that contain the term. Since the postings method only needs to know whether a document contains a given term (and not the frequency of appearance within the document), it is appropriate for a DNF representation. Figure 5.1 depicts an example of the computation of the postings. The top 10 documents are shown in the top of the figure. The relevant documents are those ones which are inside a box. A table with the value of the posting for each term is shown at the bottom of the figure. Unlike classical approaches, we apply postings to unrelevant documents as well, i.e. the number of unrelevant documents within the retrieved documents that contain the term. Specifically, positive and negative terms are selected as follows. The set of terms appearing in the retrieved relevant documents is sorted in decreasing order of postings. The top ranked terms are selected to be included in the revising formula as positive terms. In an analogous way, the set of terms appearing in the retrieved unrelevant documents is sorted in terms of postings and the top ranked terms are included as negated terms in the revising formula. In order to improve the performance of the system we only consider terms from unrelevant documents that do not appear in any relevant document. To motivate this decision, let us consider a term t appearing in some retrieved unrelevant documents and in some retrieved relevant documents. Let us consider that t has a high posting in the unrelevant documents and is selected to revise the query as $\neg t$. The inclusion of $\neg t$ in the query penalizes the documents containing t. It seems reasonable to think that, since there are some retrieved relevant documents containing t, some other relevant documents (not retrieved in the top N) can also contain t and, thus, are moved down in the rank by $\neg t$. A side effect of this choice is that the revising formula $(t_{p1} \wedge \cdots \wedge t_{pn} \wedge \neg t_{n1} \wedge \cdots \wedge \neg t_{pm})$ is always satisfiable because a term cannot be included as positive and negative. This is because the negated terms that we introduce do not belong to any retrieved relevant document and, thus, they cannot be introduced as positive.

Top 10 Documents

$\boxed{d_1 = (a \wedge b \wedge c) \vee (d \wedge e \wedge a)}$
$d_2 = b \wedge f$
$d_3 = a \vee (d \wedge a)$
$\boxed{d_4 = e \wedge a}$
$d_5 = (a \wedge b) \vee (d \wedge g)$
$d_6 = (a \wedge b) \vee (g \wedge h)$
$\boxed{d_7 = i \vee (d \wedge e \wedge a)}$
$d_8 = j \vee (k \wedge a)$
$\boxed{d_9 = a \wedge b \wedge e}$
$d_{10} = d \wedge e \wedge a$

	a	b	c	d	e	f	g	h	i	j	k
Posting	4	2	1	2	4	0	0	0	1	0	0

Posting for each term

Figure 5.1: An example of the computation of postings for relevant documents

Negative terms

Classical models use to disregard the importance of negative feedback. Term selection techniques only operate on retrieved relevant documents. Classical methods, like Rocchio and Ide-Dec-Hi approaches, consider information from unrelevant documents but in a restricted way. For instance, in the vector space model no new terms are actually added with negative weights. To motivate the importance of negative terms, let us consider a term t appearing in all the retrieved unrelevant documents and in no one retrieved relevant document. It seems reasonable to think that t is a good term to characterize the set of unrelevant documents. In the logical model we can incorporate $\neg t$ to the original query. On the contrary, the vector space model cannot use information from t because t would have a negative weight in the new query and negative weights are considered as 0 weights. We believe that negative terms can be used as a good precision-oriented mechanism.

The probabilistic model has no provision for query expansion. Harper and van Rijsbergen [26] proposed an extension that permits expansion but only positive terms are used for expansion. In the context of filtering, some works have tried to introduce a non-relevant information profile but these approaches often lack an homogeneous framework. Hoashi and other researchers [28] have recently claimed the necessity of handling non-relevant information profiles. Many systems which handle only relevant information profiles have to become conservative to avoid mistaken retrieval of non-relevant documents. This leads to set high thresholds which results in the ignorance of many relevant documents. If we are able to handle two profiles we can separate the tasks of a) retrieving relevant documents and b) avoiding non-relevant documents. In fact, these tasks are intrinsically different and they should not be treated by the same profile. A non-relevant information profile allows to reduce the number of mistakenly retrieved documents and, thus, the system can retrieve more relevant documents which may be ignored due to a conservative similarity threshold. The approach followed by Hoashi and his collaborators [28] is based on applying the non-relevant information profile on the top of the relevant information profile, i.e.

the documents which have passed the relevant information profile are filtered by the non-relevant information profile. We believe that this kind of extensions should not be accomplished in an *ad-hoc* way but within the grounds of a formal framework. In this sense, our proposal is general because we can handle both positive and negative information in an homogeneous way.

In next section we present the evaluation results of both the document oriented approach (equation 5.5) and the term oriented approach (equation 5.6). The aim of these tests is:

- clarify whether a term selection technique outperforms a direct revision with the disjunction of the retrieved relevant documents.
- analyze whether the use of negative terms produces significant improvements.

5.2 Evaluation

First we present the experimental results of the two main feedback approaches presented above. Then, we depict some variations that we tried out in the evaluation of feedback strategies.

5.2.1 Results

A simple approach to evaluate feedback strategies is to rank the set of documents using the new query and to measure precision-recall figures relative to the set of relevant documents for the original query. This evaluation strategy generally produces spectacular improvements. Unfortunately, a significant part of this improvement comes from the higher ranks assigned to the set of documents already identified as relevant by the user during the feedback process. Then, this strategy masks the real gains in performance due to documents not seen by the user yet. A more realistic approach is to evaluate the retrieval performance of the new query considering the set of all documents minus the set of documents already analyzed by the user. This set of documents is called the *residual collection*. Precision/recall results use to be worse because highly ranked documents are removed. This methodology of residual evaluation was originally proposed by Salton [52] and, now, it is widely used to compare distinct relevance feedback strategies.

To evaluate relevance feedback, interaction with the user is needed. However, a methodology largely applied is to use some standard IR collection and to approximate user's relevance judgments by collection's relevance judgments. This is not a problem because collection's judgments are provided by experts.

In this section we present the evaluation results of the two main feedback strategies presented in the previous section. We have tested the document oriented approach, which uses the disjunction of the retrieved relevant documents as the revising formula (equation 5.5), and the term oriented approach, which builds a revising formula with selected terms (equation 5.6). Tests with only positive terms and with both positive and negative terms have been done for the second approach. We have utilized four document collections, namely CACM, CRAN, CISI and LISA. Documents and queries are initially indexed as DNF formulas. A first retrieval was done using the original queries and the top ten documents were used for relevance feedback. Each document in the top 10 is marked as relevant or non-relevant using the collection's relevance judgments. Not all the original queries can be used for feedback evaluation because some of them have all their relevant documents in the top ten (and, thus, the residual collection has not any relevant document) and some of them have no relevant documents in the top ten (and, thus, we do not have positive information to revise the query). Table 5.1 presents an overview of the number of queries considered for each collection.

	CACM	CRAN	CISI	LISA
Total number of queries	52	225	76	35
Queries with all relevants in top ten	1	14	1	0
Queries with no relevants in top ten	4	21	15	16
Number of queries considered	47	190	60	19

Table 5.1: Number of queries considered for each collection

Tables 5.3, 5.4, 5.5 and 5.6 present the recall-precision results for the four document collections and figures 5.2, 5.3, 5.4 and 5.5 shows the corresponding precision versus recall graphs. The first column of each table shows the base residual run, i.e. the initial run without feedback and with the top ten documents removed. The second column presents the precision-recall values for the document oriented approach. Significant improvements are obtained when feedback is introduced. In terms of average precision, the document oriented approach improved the base residual in all the tests. The run on the CISI collection presented the poorest improvement (5.2 % in terms of average precision). This might be due to the fact that CISI has many relevant documents per query (see table 5.2) and, then, it is difficult to approximate the query to that big set of documents. A big set of documents is likely more heterogeneous. If a query has many relevant documents, the similarity among them decreases and the impact of feedback is reduced. Besides, CISI has big queries and, then, there is not much room for improvement. An important circumstance is that the largest improvements are obtained at low recall levels. Although the improvement with regard to the base residual is small, in appendix A we show that the difference is statistically significant.

	CACM	CRAN	CISI	LISA
Avg number of terms per document	35.86	96.27	66.26	49.48
Avg number of terms per query	12.61	9.48	30.31	32.21
Avg number of relevant documents per query	16.53	8.78	47.03	14.89

Table 5.2: Average number of terms in documents and queries

The third and fourth columns of each table show the results obtained when term selection is applied. We tried different values for the number of selected terms and we show the best run for each collection. Our intention is not to obtain an ideal and generically applicable value for the number of revising terms but to show that a framework modeling both positive and negative terms is desirable. In general, term selection performed better than the document oriented approach. The selection only with positive terms outperformed the document oriented approach in three collections. Only in the CISI collection the document oriented approach was better than the selection only with positives. Again, the intrinsic difficulty of this collection might have produced this situation.

The selection of both positive and negative terms presented very good performance. It was the best approach in all collections. The improvements over the approach that only selects positive terms are quite significant. It is especially attractive the case of the CISI collection, in which the selection of positives did not outperform the document oriented approach but, on the

Recall - Precision	Base Residual	Document oriented Approach	Term selection only positives Best run: 4 pos.	Term selection positives and negatives Best run: 4 pos. & 3 neg
0.00	0.3204	0.3987	0.3755	0.4215
0.10	0.2544	0.2850	0.3112	0.3513
0.20	0.1895	0.1911	0.2122	0.2370
0.30	0.1506	0.1410	0.1734	0.1951
0.40	0.1025	0.1023	0.1086	0.1350
0.50	0.0843	0.0814	0.0972	0.1141
0.60	0.0696	0.0707	0.0722	0.0912
0.70	0.0505	0.0520	0.0532	0.0576
0.80	0.0396	0.0402	0.0411	0.0454
0.90	0.0221	0.0212	0.0193	0.0232
1.00	0.0160	0.0163	0.0146	0.0171
Avg. prec.	0.1181	0.1273	0.1344	0.1535
% Prec. change		+7.7 %	+13.8 %	+20.6 %
Avg. prec. for 3 intermediate points	0.1045	0.1043	0.1168	0.1322
% Prec. change		-0.2 %	+11.8 %	+26.5 %

Table 5.3: Evaluation results for CACM collection

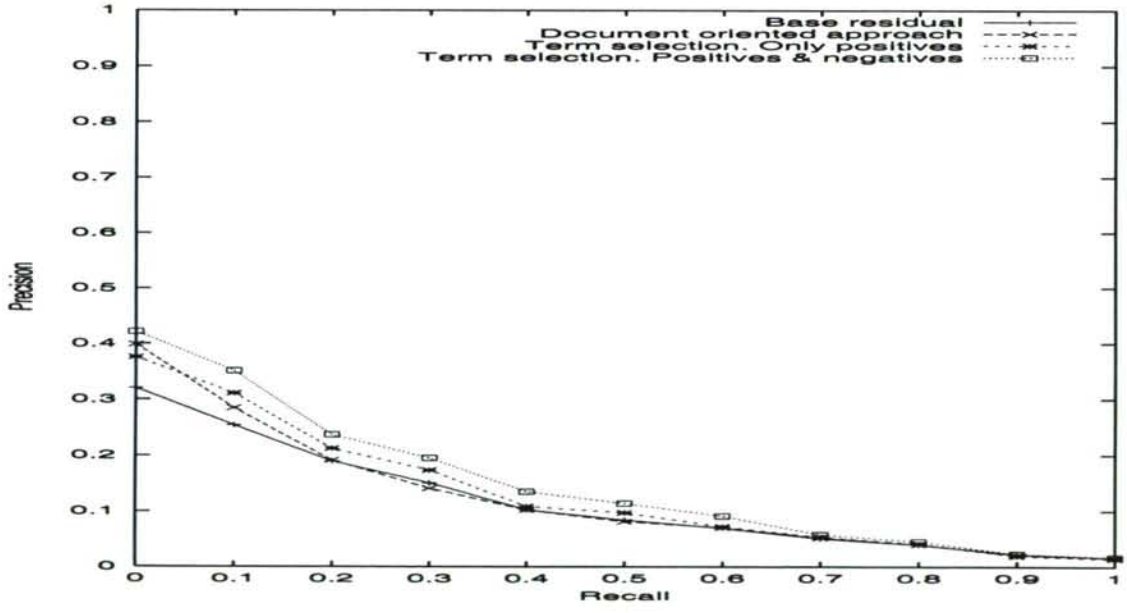

Figure 5.2: CACM precision vs. recall graph

Recall - Precision	Base Residual	Document oriented Approach	Term selection only positives Best run: 10 pos.	Term selection positives and negatives Best run: 10 pos.& 20 neg.
0.00	0.3193	0.4180	0.4075	0.4626
0.10	0.2979	0.3817	0.3825	0.4381
0.20	0.2360	0.3241	0.3432	0.3633
0.30	0.1819	0.2573	0.2538	0.2923
0.40	0.1440	0.2049	0.2026	0.2274
0.50	0.1276	0.1788	0.1819	0.2145
0.60	0.0836	0.1131	0.1298	0.1445
0.70	0.0596	0.0880	0.0918	0.1020
0.80	0.0498	0.0727	0.0797	0.0834
0.90	0.0418	0.0627	0.0722	0.0707
1.00	0.0405	0.0608	0.0707	0.0700
Avg. prec.	0.1438	0.1966	0.2014	0.2244
% Prec. change		+36.7 %	+40.1 %	+56.1 %
Avg. prec. for 3 intermediate points	0.1378	0.1919	0.2016	0.2204
% Prec. change		+33.4 %	+46.3 %	+59.9 %

Table 5.4: Evaluation results for CRAN collection

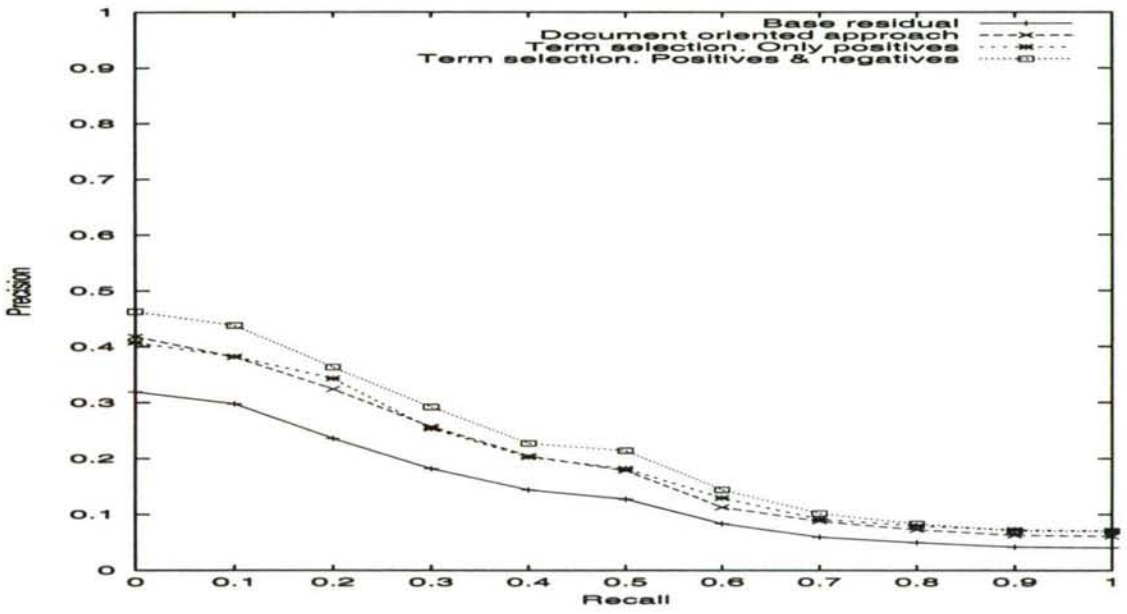

Figure 5.3: CRAN precision vs. recall graph

Recall - Precision	Base Residual	Document oriented Approach	Term selection only positives Best run: 3 pos.	Term selection positives and negatives Best run: 3 pos.& 6 neg.
0.00	0.4119	0.4650	0.4128	0.5233
0.10	0.2012	0.2130	0.2205	0.2417
0.20	0.1611	0.1648	0.1684	0.1829
0.30	0.1361	0.1392	0.1376	0.1470
0.40	0.1153	0.1171	0.1171	0.1192
0.50	0.1022	0.1015	0.1034	0.1021
0.60	0.0860	0.0873	0.0881	0.0850
0.70	0.0726	0.0737	0.0747	0.0691
0.80	0.0631	0.0631	0.0646	0.0565
0.90	0.0503	0.0496	0.0499	0.0434
1.00	0.0384	0.0382	0.0387	0.0345
Avg. prec.	0.1307	0.1375	0.1342	0.1459
% Prec. change		+5.2 %	+2.6 %	+11.6 %
Avg. prec. for 3 intermediate points	0.1088	0.1098	0.1122	0.1138
% Prec. change		+0.9 %	+3.1 %	+4.6 %

Table 5.5: Evaluation results for CISI collection

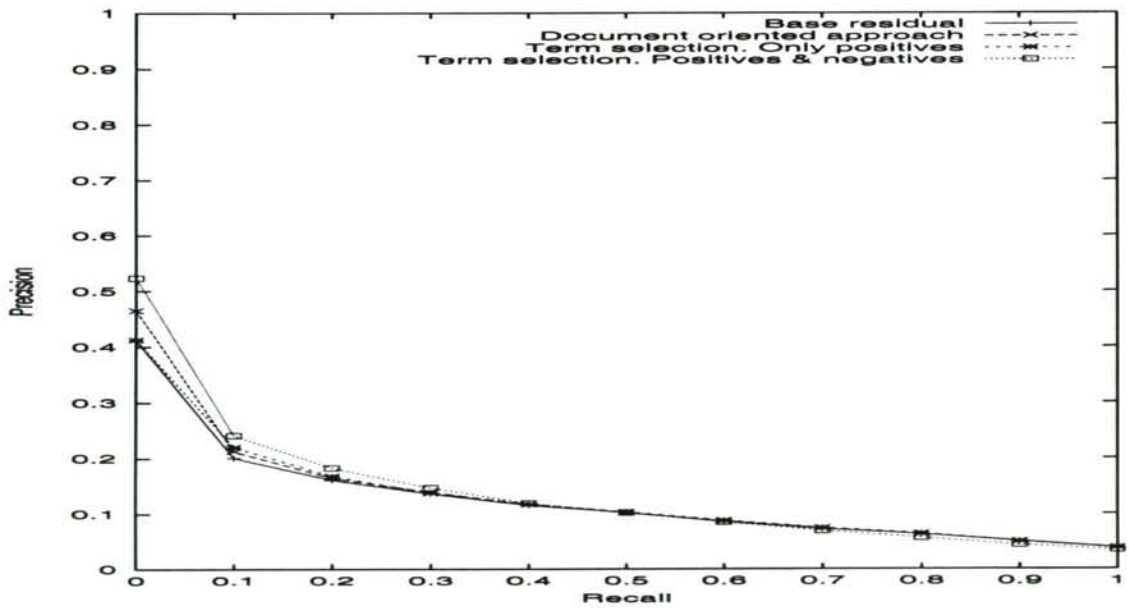

Figure 5.4: CISI precision vs. recall graph

Recall - Precision	Base Residual	Document oriented Approach	Term selection only positives Best run: 14 pos.	Term selection positives and negatives Best run: 14 pos.& 40 neg.
0.00	0.1266	0.1934	0.2411	0.2204
0.10	0.0598	0.1111	0.1202	0.1794
0.20	0.0468	0.0532	0.0453	0.0752
0.30	0.0204	0.0199	0.0180	0.0144
0.40	0.0120	0.0113	0.0131	0.0082
0.50	0.0086	0.0081	0.0096	0.0063
0.60	0.0058	0.0059	0.0072	0.0051
0.70	0.0049	0.0050	0.0054	0.0037
0.80	0.0038	0.0039	0.0037	0.0032
0.90	0.0029	0.0029	0.0031	0.0027
1.00	0.0026	0.0026	0.0027	0.0026
Avg. prec.	0.0268	0.0379	0.0427	0.0474
% Prec. change		+41.4 %	+59.3 %	+76.9 %
Avg. prec. for 3 intermediate points	0.0198	0.0217	0.0195	0.0283
% Precision change		+9.6 %	-1.5 %	+42.9 %

Table 5.6: Evaluation results for LISA collection

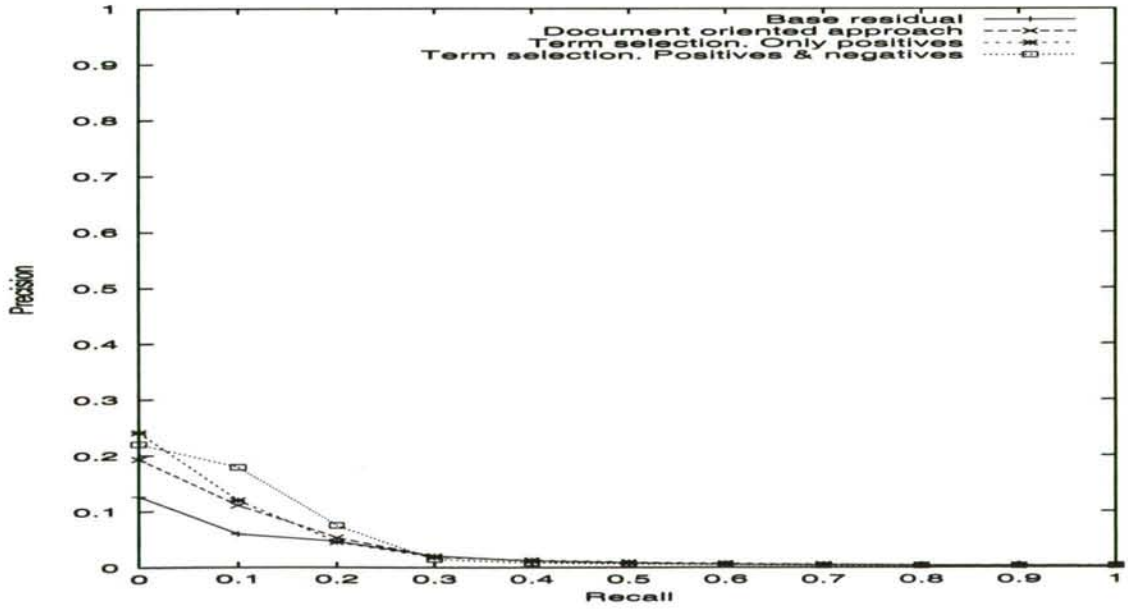

Figure 5.5: LISA precision vs. recall graph

contrary, the selection of both positives and negatives was significantly better than the document oriented approach. In this sort of environments (like CISI), where it is hard to approximate the query to the set of relevant documents, it becomes especially useful to have a method to move the query away from unrelevant documents.

5.2.2 Additional experiments

We have tried out some variations in the model of feedback but no one improved performance. Kindo and other researchers [33] designed an adaptative learning algorithm in the context of information filtering systems. This algorithm is based on a term classifier which is able to classify terms into three groups. The first group includes those terms whereby if one of them appears in an item, then the user has an interest in the item. These terms are called *positive keywords. Negative keywords* are those terms whereby if one of them appears in an item, then the user is not interested in the item. Finally, *neutral keywords* are those which are not useful for determining whether or not the user has an interest in the items containing the term. Intuitively, positive keywords represent the information items in which the user is interested, negative keywords represent information items that the user does not want to retrieve and neutral keywords are keywords not useful for determining user's interests. The keyword classifier manipulates documents, keywords and user's inputs through statistical methods. The next equation was developed by Lau and his colleagues [36] based on the keyword classifier designed by Kindo et al [33].

$$pre(k) = \epsilon \times tanh(\frac{df(k)}{\mathcal{E}}) \times (p(k_{rel})tanh\frac{p(k_{rel})}{p_{rel}} - (1 - p(k_{rel}))tanh\frac{(1 - p(k_{rel}))}{(1 - p_{rel})}) \quad (5.7)$$

This equation can be used to induce the preference value $pre(k)$ of a keyword k and then classify it as positive, negative or neutral. In the equation ϵ is used to restrict the range of $pre(k)$ such that it holds that $-1 < pre(k) < 1$. The value of $df(k_{rel})$ is the number of relevant documents that contains the keyword k (i.e. the posting of k) and $df(k_{nrel})$ is the number of non-relevant documents that contains the keyword k. The value of $df(k)$ is the sum of $df(k_{rel})$ and $df(k_{nrel})$ and $tanh$ is the hyperbolic tangent. The parameter $\mathcal{E}$ is used to control rare or new keywords and is expressed as $int(logN + 1)$, where N is the total number of documents which have been judged by the user. The function int is an integer function which truncates the decimal values. $p(k_{rel})$ is the estimated probability that a document containing the keyword k is relevant. The value of $p(k_{rel})$ is expressed as the fraction $\frac{df(k_{rel})}{df(k_{rel})+df(k_{nrel})}$. p_{rel} is the estimated probability that a document is relevant. It can be assumed that $p_{rel} = 0.5$.

A positive value of $pre(k)$ implies that the associate keyword is positive, whereas a negative value of $pre(k)$ indicates a negative keyword. A threshold λ is selected (typically 0.5) and if the absolute value of $pre(k)$ is below $lambda$, the keyword is considered neutral.

This method was used by Lau and other researchers [36] in the context of Information Filtering agents. Information filtering agents are computer systems that automatically filter an incoming stream of information on behalf of the users. User's information needs change over time and, thus, information filtering agents must be able to revise their beliefs about the user's information needs. Lau and his collaborators proposed to formalize agent's beliefs by theories in a propositional language. In order to update agent's beliefs, the notion of *epistemic entrenchment* $(\leq)$ is used. If α and β are beliefs in a belief set, $\alpha \leq \beta$ means that β is at least as entrenched as α. If inconsistency arises after applying changes to a belief set, beliefs with the lowest degree of epistemic entrenchment are given up. The output of the keyword classifier developed by Kindo

et al.[33] was applied to build an epistemic entrenchment. Positive keywords are selected to revise the original belief state as positive propositional formulas, p. Negative keywords revise the belief state as negative propositional formulas, $\neg p$, whereas neutral keywords are not considered for the revision process. If inconsistency arises, the epistemic entrenchment determines the formulas that should be deleted. Keywords with high value of $|pre(k)|$ are assigned high degrees of epistemic entrenchment and, as a result, they are hardly eliminated from the belief state. On the contrary, keywords with low values of $|pre(k)|$ are assigned low degrees of epistemic entrenchment and they will be likely deleted from the belief state.

We think Lau et al.'s approach [36] is appropriate for updating agent's beliefs. Besides, they used a finite representation of the epistemic entrenchment ordering [63, 64] that is appropriate for a computer based implementation. However quantitative evaluation of the effectiveness of their proposal has not been conducted yet. Besides, they utilize the notion of logical consequence to match the filtering agent's belief state with the incoming documents and, hence, only a binary relevance decision can be made.

We have experimented with the ideas presented in the previous paragraphs within our model. Specifically, we implemented the keyword classifier developed by Kindo et al.[33] and we used it for obtaining positive and negative terms. We were able to apply this method because the keyword classifier (equation 5.7) does not utilize any measure of term frequency within documents. As we argued before, DNF representations cannot cope with this kind of measures. The update method is the same that the one we presented for the postings method but the terms are selected using the keyword classifier:

$$q_m = q \circ (t_{p1} \wedge \cdots \wedge t_{pn} \wedge \neg t_{n1} \wedge \cdots \wedge \neg t_{nm}) \tag{5.8}$$

where it holds that $\forall_{i=1...n}\ pre(t_{pi}) > \lambda$ and $\forall_{i=1...m}\ pre(t_{ni}) < -\lambda$ (recall that λ is a threshold which indicates the limit of neutral keywords). We applied this method instead of the postings method and we ran several experiments varying the parameters involved in the keyword classifier. Nevertheless, in all the tests we ran the postings method performed better than the keyword classifier. This could indicate that the keyword classifier is only useful in very dynamic environments like information filtering systems.

We also tried out other extensions with regard to the form of the revising formula of equation 5.6. Basically, instead of using a conjunction of positive and negative terms, we built generic DNF formulas. To get a DNF formula we computed *clause-based* postings, i.e. we computed postings within clauses and the best terms for each clause are represented in a clause of the revising formula. Again, this technique did not improve performance. This might be due to the small size of most of the subfields of the documents. Only the subfield W contains a large number of terms and, thus, the document has one big clause and several small ones. If documents have subfields with more homogeneous sizes, this technique might perform better.

5.2.3 Final remarks

The conclusions of the experiments presented here are clear. A framework allowing both positive and negative terms is desirable. The approach that handles positive and negative terms was the best one in all the tests we run. The evaluation is general enough because it was done on several (and heterogeneous) document collections and a considerable number of queries was tested. As a matter of fact, the CRAN collection has been widely used for evaluating feedback

Recall - Precision	Harman	Our model
0.00	0.5280	0.4626
0.10	0.5040	0.4381
0.20	0.4580	0.3633
0.30	0.3750	0.2923
0.40	0.3410	0.2274
0.50	0.3110	0.2145
0.60	0.2200	0.1445
0.70	0.1890	0.1020
0.80	0.1600	0.0834
0.90	0.1340	0.0707
1.00	0.1320	0.0700
Avg. prec. for 3 intermediate points	0.3090	0.2204

Table 5.7: Feedback within a classical model vs. Feedback within our model

strategies [24, 25]. The large number of queries in this collection is a good property for evaluating feedback strategies. Table 5.7 presents the precision/recall figures of our best run against CRAN and Harman's best run [25] against the same collection. Harman used a variation of the probabilistic model that combines probabilistic term reweighting and weighting using document term frequencies. Since there is no built-in query expansion using the probabilistic model, Harman applied different techniques to expand the query with new terms. The best run applies both term reweighting and query expansion. This does not pretends to be a exact comparison because Harman's work uses 196 queries and we used 190 but we just want to illustrate that our *binary-weighted* model is not very far from classical models that use non-binary term frequencies within documents. Note that the difference is even closer at low recall levels.

Chapter 6

Incorporating term similarity and inverse document frequency into the model

The model presented in chapter 2 uses distances between interpretations which are computed within a revision process. We proposed Dalal's revision operator because it establishes an order between interpretations which corresponds with the ranking imposed by some classical IR matching functions. A measure of distance between two interpretations was determined by the number of propositional letters in which the interpretations differ. Nevertheless, since we are defining a model for IR, we could take benefit from additional information which is peculiar to this application domain. Let us recall that the propositional alphabet models the indexing vocabulary of an IR system. The measure of distance between interpretations that we have utilized so far considers that all the propositional letters are equally important. Given two interpretations, each *non-matching* propositional letter (i.e. a propositional letter mapped into true by one interpretation and mapped into false by the other interpretation) adds 1 to the distance between the interpretations, no matter which the letter is. In an analogous way, *matching* letters (i.e. those propositional letters mapped into the same truth value by the interpretations) add 0 to the distance between the interpretations, no matter which these letters are. However, IR models often apply useful intuitions to measure the relative importance of the keywords involved in the matching process. Some IR models define ways to incorporate information from non-matching terms at retrieval time. For instance, a query asking for documents about `informatics` might not retrieve a document not mentioning `informatics` but mentioning `computers`. Clearly, this sort of documents seems to be relevant to the user. If we articulate a method that informs us that computers and informatics are related concepts, we would be able to make a better relevance decision. A number of measures of similarity between terms have been proposed in the literature to overcome this problem. Non-matching terms are not considered equally bad but, instead, they are measured using term similarity information. On the other hand, matching keywords between document's and query's representations are usually weighted depending on factors such as the frequency of appearance of the keyword within the whole collection. Intuitively, a term appearing in most of the documents is not very helpful to discriminate between relevant and unrelevant documents. On the contrary, a term which appears in few documents has a good chance of being a proper representative of the contents of the documents in which it appears. The notion of *inverse document frequency* (idf) formalizes this intuition. The idf factor often

plays an important role in classical matching processes.

In this chapter we define a new similarity measure which takes into account both similarity between terms and inverse document frequency. The start point is the definition of a new measure of distance between a given interpretation and a set of interpretations. Now, common terms between a document and a query do not simply add 0 to the measure of distance between the document and the query but produce an increment on the distance which depends on idf information. The frequency of appearance of these terms within the whole collection determines the final value of this distance. On the other hand, non-matching terms will produce an increment to the distance which depends on a measure of similarity between terms.

The reader might wonder why we do not incorporate the *term frequency* (tf) factor to the model. The idf factor and a measure of similarity between terms are *global* notions, i.e. they do not depend on a particular document but are characteristics of the whole collection (furthermore, the notion of term similarity is not collection-dependent because we can get a measure of similarity between terms from a thesaurus, from other collections, etc.). These notions help us to refine the matching process because they introduce additional information about the involved terms. However, our representational formalism keeps being the same: Propositional Logic. Documents and queries are propositional formulas as before. The tf factor, which is determined by the number of occurrences of a term within a document, is not a global notion but it is associated to a particular document. At matching time, we can use the idf factor and term similarity information for measuring the distance between two interpretations because they are global factors and, hence, we do not need to know which document/query is being handled. On the contrary, to apply the tf factor we would need to know which document/query corresponds to the interpretations being handled. This would not be possible because a given interpretation can be a model of many documents and queries. Hence, the notion of interpretation would have to incorporate term frequencies giving rise to a totally different model. As a result, an homogeneous extension of the model cannot consider term frequency information. Nevertheless, the formalism allows to model the notion of significance of a term in a document through DNF formulas with several clauses. Distinct clauses can represent distinct parts of a document and, given a term, we can express in which parts of the document it appears through the inclusion of the term in the corresponding clauses. Besides, this is reflected in the measure of similarity.

The rest of the chapter is organized as follows. First, we present the model of matching that includes term similarity information. Second, we extend the model to deal with idf information. New experiments and evaluation results are presented in the last part of the chapter.

6.1 Incorporating term similarity information

In order to extend the model to deal with term similarity information we define a new order between interpretations. Along this chapter, we focus on the extension of the exhaustivity-oriented measure $BRsim$, which is based on the BR process $q \circ_D d$.

Let us recall that Dalal's revision operator, $\circ_D$, uses the number of propositional letters on which two interpretations I and J differ as a measure of *distance* between them, $dist(I, J)$. Since we are representing interpretations by the set of letters mapped into true, a measure of distance between two interpretations can be directly obtained from the cardinality of the symmetric difference between their respective sets. A measure of distance between the set of models of a formula ψ and a given interpretation I was defined as the distance from I to its closest interpretation in $Mod(\psi)$:

$$Dist(I, Mod(\psi)) = \min_{M \in Mod(\psi)} dist(M, I) \tag{6.1}$$

This distance was used to define the measure $BRsim$ as the average of the distances from each model of the document to the set of models of the query.

We think that it is useful to refine the previous definitions taking into account the concrete domain of application. Basically, not all differing terms are equally bad. Consider fig. 6.1 where we depict an example of the computation of the distances applied so far. The model of the document is assigned a distance of 1 to the set of models of the query because the letter b is mapped into false by the model of the document and it is mapped into true by all the models of the query. Intuitively, the query is asking for documents about b and the document does not mention the term b. However, it can be the case that either a or c, which are terms mentioned by the document, are closely related to b. In that case, it seems reasonable to measure the distance from the model of the document to the query taking into account the similarity between the term b and the terms a and c. If a and c are not related to b then we should take a distance of 1 as before but if either a or c are similar to b, the value of the distance should be fixed accordingly.

However, not all differing terms between documents and queries should be treated in the same way. Consider fig. 6.2 where a new example is presented. The example is analogous to the one presented before but the negation of the term b was interchanged. The final value of the distance is the same because we keep having the same differing term, b. However, in this case the query is asking for documents not dealing with b and the document mentions the term b. It seems reasonable to think that, now, the use of term similarity information does not make sense.

Following these intuitions we propose to classify differing terms depending on whether they appear as positive or as negative literals in the query. Differing terms appearing negated in the query should add 1 to the final value of distance whereas positive query terms should increase the value of distance depending on their similarity to the terms appearing in the document. All these considerations lead to the next definitions.

A new measure of distance from a given interpretation I to the set of models of a formula ψ is defined. This distance will later be used to measure the closeness from models of a document to the set of models of a query. First, we collect the smallest symmetric differences (SSD) between a given interpretation I and the set of models of a formula ψ. This set consists of the sets of differing letters between I and models of ψ having the smallest cardinality.

$$SSD(I, Mod(\psi)) = \{I \triangle J | J \in Mod(\psi) \wedge \nexists J' \in Mod(\psi), J' \neq J, |I \triangle J'| < |I \triangle J|\}, \tag{6.2}$$

where $\triangle$ is the symmetric difference between two sets. Let us recall that, given two sets A and B, the symmetric difference between them, $A \triangle B$, is given by $(A \cup B) \setminus (A \cap B)$, where $\setminus$ is the regular difference between sets.

All elements belonging to SSD have the same cardinality. Dalal's operator just takes this value of cardinality to measure the distance from I to $Mod(\psi)$. We refine Dalal's approach as follows.

Given the set $SSD(I, Mod(\psi))$, each set $sd \in SSD(I, Mod(\psi))$ can be divided into two complementary sets as follows:

$$P(sd, I) = \{l \in sd | l \in I\} \tag{6.3}$$

$$N(sd, I) = \{l \in sd | l \notin I\} \tag{6.4}$$

$\mathcal{P}= \{a, b, c\}$
$d = a \wedge \neg b \wedge c$
$q = b$

doc models → query models↓	d $\{a, c\}$
$\{a, b\}$	$\{b, c\}$
$\{a, b, c\}$	$\{b\}$
$\{b\}$	$\{a, b, c\}$
$\{b, c\}$	$\{a, b\}$

(6.1.a) Symmetric differences between query and document models

doc models → query models↓	d $\{a, c\}$
$\{a, b\}$	2
$\{a, b, c\}$	1
$\{b\}$	3
$\{b, c\}$	2

(6.1.b) Cardinalities of symmetric differences

$$Dist(m_d, Mod(q)) = min\{2, 1, 3, 2\} = 1$$

Figure 6.1: Distance from a model of the document to the set of models of the query

$\mathcal{P}= \{a, b, c\}$
$d = a \wedge b \wedge c$
$q = \neg b$

doc models → query models↓	d $\{a, b, c\}$
$\{a\}$	$\{b, c\}$
$\{a, c\}$	$\{b\}$
$\emptyset$	$\{a, b, c\}$
$\{c\}$	$\{a, b\}$

(6.2.a) Symmetric differences between query and document models

doc models → query models↓	d $\{a, b, c\}$
$\{a\}$	2
$\{a, c\}$	1
$\emptyset$	3
$\{c\}$	2

(6.2.b) Cardinalities of symmetric differences

$$Dist(m_d, Mod(q)) = min\{2, 1, 3, 2\} = 1$$

Figure 6.2: Other example of the distance from a model of the document to the set of models of the query

That is, the set $P(sd, I)$ contains the propositional letters which are mapped into true by the interpretation I (and, thus, they are mapped into false by the interpretation $J \in Mod(\psi)$ such that $sd = I \triangle J$). Analogously, the set $N(sd, I)$ contains the propositional letters which are mapped into false by the interpretation I (and, thus, they are mapped into true by the interpretation $J \in Mod(\psi)$ such that $sd = I \triangle J$). Letters should receive a different treatment depending on whether they belong to $P(sd, I)$ or $N(sd, I)$. Letters which belong to $N(sd, I)$ are mapped by the document's interpretation (I) into false and they are mapped by the associated query's interpretation into true. Consider a query asking for a term which does not appear in the document. It makes sense to use term similarity information in order to determine whether the document contains other terms which are similar to the one the query is asking for. Letters which belong to $P(sd, I)$ are mapped by the document's interpretation (I) into true and they are mapped by the associated query's interpretation into false. That is, these terms are mentioned by the document but the interpretation of the query does not want to retrieve documents containing these letters. In this case it does not make sense to use term similarity information and these terms should directly increase the value of distance as Dalal's approach does.

Let us consider that we have a measure of similarity between terms $tsim : \mathcal{P} \times \mathcal{P} \rightarrow [0, 1]$, where $\mathcal{P}$ is the propositional alphabet. Given two terms t_1 and t_2, a value of $tsim$ near to 1 means that t_1 and t_2 are closely related, whereas a value of $tsim$ near to 0 means that t_1 and t_2 are not related at all. The function $tsim$ is assumed to be symmetric. Given a propositional letter t, a term similarity function $tsim$ and a set of propositional letters A, the next formula chooses the highest similarity between t and terms belonging to A:

$$mxs(t, A) = \begin{cases} 0 & \text{if } A = \emptyset \\ \max_{t_i \in A}(tsim(t, t_i)) & \text{otherwise} \end{cases} \tag{6.5}$$

We propose now to measure the distance from an interpretation I to the set of models of a formula ψ as follows.

$$Dist(I, Mod(\psi)) = \min_{sd \in SSD(I, Mod(\psi))} (|P(sd, I)| + \sum_{l \in N(sd, I)} (1 - mxs(l, I))), \tag{6.6}$$

That is, differing terms which are negated by I produce an increment to the distance which depends on their similarity to terms mapped into true by I ($mxs(l, I)$, recall that I contains the terms mapped into true by the interpretation I). Note that the definition of the distance from I to $Mod(\psi)$ used by Dalal's operator can be rewritten using the previous definitions:

$$Dist(I, Mod(\psi)) = \min_{sd \in SSD(I, Mod(\psi))} (|P(sd, I)| + |N(sd, I)|) \tag{6.7}$$

Indeed, in the case of Dalal's distance, to extract the minimum is not needed because all $sd \in SSD(I, Mod(\psi))$ have the same cardinality and, hence, the same value of $|P(sd, I)| + |N(sd, I)|$.

Given a term similarity function $tsim$, fig. 6.3 shows an example of the computation of the new value of distance. The example assumes some values of $tsim$ needed to develop the process. The differing term b adds 1 to the distance because it is a term negated in the query. On the other hand, the increment coming from the differing term a is not 1 because, although the document does not deal with a, it deals with d, which is a term closely related to a ($tsim(a, d) = 0.8$).

$\mathcal{P} = \{a, b, c, d\}$
$d = \neg a \wedge b \wedge c \wedge d$
$q = a \wedge \neg b \wedge c$

$tsim(a, b) = 0.1$
$tsim(a, c) = 0.4$
$tsim(a, d) = 0.8$

doc models → query models↓	m_d $\{b, c, d\}$
$\{a, c\}$	$\{a, b, d\}$
$\{a, c, d\}$	$\{a, b\}$

(6.3.a) Symmetric differences between query and document models

$SSD(m_d, Mod(q)) = \{\{a, b\}\}$
$sd_1 = \{a, b\}$
$P(sd_1, m_d) = \{b\}$
$N(sd_1, m_d) = \{a\}$
$Dist(m_d, Mod(q)) = |\{b\}| + (1 - mxs(a, m_d))$
$mxs(a, m_d) = mxs(a, \{b, c, d\}) = max\{0.1, 0.4, 0.8\} = 0.8$
$Dist(m_d, Mod(q)) = 1 + (1 - 0.8) = 1.2$

Figure 6.3: New distance from a model of the document to the set of models of the query

Note that Dalal's approach would take $|P(sd_1, m_d)| + |N(sd_1, m_d)| = 2$ to measure the distance from m_d to $Mod(q)$.

So far, we have defined a distance from a given interpretation I to the set of models of a formula ψ. From this definition we can now formalize a new total preorder between interpretations. As before, given a formula ψ, a total preorder between interpretations can be extracted from the nearness of each interpretation to the set of models of the formula ψ. Formally, a total preorder $\leq_\psi$ is defined as:

$$I \leq_\psi J \quad \text{iff} \quad Dist(I, Mod(\psi)) \leq Dist(J, Mod(\psi)) \tag{6.8}$$

Intuitively, $I \leq_\psi J$ means that I is closer to $Mod(\psi)$ than J but, now, the notion of closeness takes into account term similarity information. Again, we can define $<$ as $I < J$ if and only if $I \leq J$ and $J \not\leq I$.

Following this order we can define a new revision operator. When revising a theory ψ with a new information μ, the models of the new information which are the closest to the theory (w.r.t the new order induced by the theory) are selected to be the models of the revised theory:

$$Mod(\psi \circ \mu) = Min(Mod(\mu), \leq_\psi) \tag{6.9}$$

At this point we are not interested in applying this operator for revising a theory. As in chapter 2 our aim is to use the distances between interpretations to get a non-binary measure of the entailment $d \models q$. Nevertheless, it is interesting to check whether the assignment defined in equation 6.8 is faithful. Let us recall that if the assignment is faithful, the revision process fulfills

the BR postulates. This assures a *rational* revision which could be applied in future extensions of the model.

The assignment defined in equation 6.8 is faithful if the following three conditions hold:

1. If $I, I' \in Mod(\psi)$, then $I <_\psi I'$ does not hold.
2. If $I \in Mod(\psi)$ and $I' \notin Mod(\psi)$, then $I <_\psi I'$ holds.
3. If $\psi \equiv \phi$, then $\leq_\psi = \leq_\phi$.

In the next lines we study whether or not these properties are satisfied for the current assignment.

1. Let I, I' be two models of ψ. Since $I, I' \in Mod(\psi)$, $SSD(I, Mod(\psi)) = \{\emptyset\}$ and $SSD(I', Mod(\psi)) = \{\emptyset\}$. As a result, $Dist(I, Mod(\psi)) = Dist(I', Mod(\psi)) = 0$ and, hence, $I <_\psi I'$ does not hold because both $I \leq_\psi I'$ and $I' \leq_\psi I$ hold.
2. Let I be a model of ψ and I' a non-model of ψ. Again, it holds that $Dist(I, Mod(\psi)) = 0$. I' is not a model of ψ and, thus, $\emptyset$ does not belong to $SSD(I', Mod(\psi))$. Consider every $sd \in SSD(I', Mod(\psi))$. If $|P(sd, I')| > 0$ it follows directly that $Dist(I', Mod(\psi)) > 0$ and, hence, $I <_\psi I'$. On the other hand if $P(sd, I') = \emptyset$ then it holds that $N(sd, I')$ is not empty. Then, the value of $Dist(I', Mod(\psi))$ is equal to $\sum_{t \in N(sd, I')}(1 - mxs(t, I'))$. If we assure that the term similarity function is only equal to 1 if their arguments are the same then each $(1 - mxs(t, I'))$ is strictly greater than zero (note that $t \notin I'$ because $t \in N(sd, I')$) and, hence, $\sum_{t \in N(sd, I')}(1 - mxs(t, I'))$ is strictly greater than 0. Note that, in the case that I' maps every propositional letter into false (i.e. $I' = \emptyset$), $(1 - mxs(t, I'))$ is greater than 0 (indeed $(1 - mxs(t, I')) = 1$). Hence, to assure that condition 2 holds we need a term similarity function $tsim$ such that $tsim(t_1, t_2) = 1$ implies that $t_1 = t_2$.
3. If $\psi \equiv \phi$, $Mod(\psi) = Mod(\phi)$ and, thus, it follows directly that $\leq_\psi = \leq_\phi$.

As a result, the assignment is faithful as long as the term similarity function fulfills a basic condition. This condition is completely acceptable: the similarity is maximum only if the terms are the same. Indeed, this property of the term similarity measure is usually accepted by IR term similarity approaches [11].

The new measure from a given interpretation I to the set of models of a formula ψ was defined to measure the distance from a model of the document to the set of models of the query. As in chapter 2, we propose to use the average of the distances from each model of the document to the set of models of the query as the distance from the document to the query:

$$distance(d, q) = \frac{\sum_{m \in Mod(d)} Dist(m, Mod(q))}{|Mod(d)|} \tag{6.10}$$

Again, a total preorder $\leq_q$ induced by the query can be defined. This preorder establishes formally a method to rank documents in terms of their respective distances to the query. Now this order takes into account term similarity information.

$$d_i \leq_q d_j \text{ iff } distance(d_i, q) \leq distance(d_j, q) \tag{6.11}$$

We can also define $<_q$ as $d_i <_q d_j$ iff $d_i \leq_q d_j$ and not $d_j \leq_q d_i$, and $=_q$ as $d_i =_q d_j$ iff $d_i \leq_q d_j$ and $d_j \leq_q d_i$.

The distance can be directly transformed into a similarity measure in the interval $[0, 1]$. The next equation defines a generalized version of the similarity measure $BRsim$, TS-BRsim which is a normalization of *distance*. It can be easily shown that, since the term similarity function takes values in the interval $[0, 1]$, the number of letters appearing in the query keeps being an upper bound for *distance*.

$$\text{TS-BRsim}(d, q) = 1 - distance(d, q)/k, \tag{6.12}$$

where k is the number of propositional letters appearing in q. Figure 6.4 shows an example of the computation of TS-BRsim for a document having several models. Some values of $tsim$ are assumed. The document has two models, $m_1 = \{a, b, c\}$ and $m_2 = \{a, c\}$. The model m_1 is also a model of the query and, hence, $Dist(m_1, Mod(q)) = 0$. On the other hand, the closest symmetric difference from the model m_2 to $Mod(q)$ is $\{b\}$. Since b is mapped into false by m_2, $Dist(m_2, Mod(q))$ is determined by the similarity between b and c, which is the term mapped into true by m_2 having the highest similarity to b.

6.2 Implementation

Again, a direct computation of the similarity measure TS-BRsim would take exponential time because of the computations involving models. In this section, we present efficient algorithms that calculate the new similarity measure. The basic technique is the same presented in chapter 3 but the use of term similarity information introduces additional considerations.

Formulas representing documents and queries follow the same syntactic characterization as before. We represent documents and queries as DNF formulas. Recall that a DNF formula is represented by a set of clauses, $\psi = \{\psi_1, \psi_2, \dots\}$, where each clause is a set of literals representing their conjunction. The set ψ represents the disjunction of all the clauses.

In order to compute the similarity measure TS-BRsim, we need to compute the average distance from models of the document to the set of models of the query. Next sections present the definitions needed to compute an approximate value to this distance. All these definitions have been built with the intention that ψ is the query and μ is the document. In an analogous way, ψ_i is a query clause and μ_j a document clause.

6.2.1 Distance between clauses

In this section we define a measure of distance between clauses. Given two clauses ψ_i and μ_j, $distance(\psi_i, \mu_j)$ computes an approximate value to the average distance from models of μ_j to the set of models of ψ_i. We use a number of definitions that manage literals syntactically appearing in ψ_i and μ_j. Later on, in the next section, we present the algorithm that computes an approximate value to TS-BRsim(μ, ψ), where μ and ψ are DNF formulas with one clause representing a document and a query respectively. This algorithm is based on the computation of the distance between the query and the document clause defined here. At that point, we will explain why this distance is an approximate value to the average distance from models of the document to the set of models of the query.

The difference between two clauses ψ_i and μ_j is defined as the set of contradicting literals:

$\mathcal{P}$= $\{a, b, c\}$
$d = a \wedge c$
$q = a \wedge b$

$tsim(b, a) = 0.1$
$tsim(b, c) = 0.6$

document models → / query models↓	m_1 $\{a, b, c\}$	m_2 $\{a, c\}$
$\{a, b\}$	$\{c\}$	$\{b, c\}$
$\{a, b, c\}$	$\emptyset$	$\{b\}$

(6.4.a)Symmetric differences between query and document models

$SSD(m_1, Mod(q)) = \{\emptyset\}$
$sd_{11} = \emptyset$
$P(sd_1, m_1) = \emptyset$
$N(sd_1, m_1) = \emptyset$
$Dist(m_1, Mod(q)) = 0$

$SSD(m_2, Mod(q)) = \{\{b\}\}$
$sd_{21} = \{b\}$
$P(sd_{21}, m_2) = \emptyset$
$N(sd_{21}, m_2) = \{b\}$
$Dist(m_2, Mod(q)) = 0 + (1 - mxs(b, m_2))$
$mxs(b, m_2) = mxs(b, \{a, c\}) = max\{0.1, 0.6\} = 0.6$
$Dist(m_2, Mod(q)) = 0 + (1 - 0.6) = 0.4$

$distance(d, q) = \frac{0+0.4}{2} = 0.2$
TS-BRsim$(d, q) = 1 - 0.2/2 = 0.9$

Figure 6.4: An example of the computation of TS-BRsim

$$CDiff(\psi_i, \mu_j) = \{l \in \psi_i | \neg l \in \mu_j\} \tag{6.13}$$

Now we need to distinguish between literals which are positive in μ_j and literals which are negative in μ_j. Thus, we divide the set $CDiff(\psi_i, \mu_j)$ into two complementary sets as follows:

$$CDiffP(\psi_i, \mu_j) = \{l \in CDiff(\mu_j, \psi_i) | l \textit{ is a positive literal}\} \tag{6.14}$$
$$CDiffN(\psi_i, \mu_j) = \{l \in CDiff(\mu_j, \psi_i) | l \textit{ is a negative literal}\} \tag{6.15}$$

That is, $CDiffP$ contains the literals that are positive in μ_j and $CDiffN$ contains the literals that are negative in μ_j.

Given a clause c, the next formula collects the letters which appear in a positive literal:

$$lpl(c) = \{l \in c | l \textit{ is a positive literal}\} \tag{6.16}$$

The function *letter* takes a literal and returns its propositional letter, e.g. $letter(\neg a) = a$.

In order to compute the average distance from models of μ_j to the set of models of ψ_i, the next formula defines the contribution to this distance from differing literals which are negated in μ_j. Basically, for each differing literal l in $CDiffN(\psi_i, \mu_j)$ we look for the highest similarity between $letter(l)$ and letters in a positive literal of μ_j ($mxs(letter(l), lpl(\mu_j))$). The more similarity between $letter(l)$ and its closest positive literal(s) in μ_j, the less increment to the distance. Intuitively, although μ_j negates the term $letter(l)$, it can have positive terms being similar to $letter(l)$.

$$CDistN(\psi_i, \mu_j) = \sum_{l \in CDiffN(\psi_i, \mu_j)} (1 - mxs(letter(l), lpl(\mu_j))), \tag{6.17}$$

On the other hand, each differing literal which is positive in μ_j increases 1 the average distance from models of μ_j to the set of models of ψ_i. This leads to the following definition of the contribution to the distance from differing literals which are positive in μ_j.

$$CDistP(\psi_i, \mu_j) = |CDiffP(\psi_i, \mu_j)| \tag{6.18}$$

So far we have defined the contributions to the distance coming from the differing literals between two clauses. Now, we define the contribution to the distance coming from literals in ψ_i whose propositional letter is not mentioned by μ_j (either in a positive or in a negative literal). Given two clauses ψ_i and μ_j, we collect the literals in ψ_i whose propositional letter is not mentioned by μ_j as follows.

$$nm(\psi_i, \mu_j) = (\psi_i \setminus \psi_i \cap \mu_j) \setminus CDiff(\psi_i, \mu_j) \tag{6.19}$$

First, literals belonging to ψ_i that also appear in μ_j are eliminated ($\psi_i \setminus \psi_i \cap \mu_j$). Second, the literals belonging to ψ_i whose opposite literal appears in μ_j are eliminated. Note that $CDiff(\psi_i, \mu_j)$ contains the differing literals as mentioned by its first argument, ψ_i, i.e.

$CDiff(\psi_i, \mu_j) \subseteq \psi_i$. One could wonder why we do not consider the reciprocal case, i.e. the literals in μ_j not mentioned by ψ_i. Note that ψ_i is a query clause and μ_j is a document clause and, hence, the literals in μ_j not mentioned by ψ_i do not increase the distance. This is because the distance from a μ_j model to the set of models of ψ_i takes the smallest symmetric differences between the μ_j model and models of ψ_i. Given a literal appearing in μ_j but not appearing in ψ_i, there is always a model of ψ_i mapping its letter into the same truth value than the models of μ_j.

Now we divide the non-mentioned literals into two complementary sets depending on whether they are positive or negative literals. Formally, given two clauses ψ_i and μ_j:

$$pnm(\psi_i, \mu_j) = \{l \in nm(\psi_i, \mu_j) | \text{l } is\ a\ positive\ literal\} \tag{6.20}$$

$$nnm(\psi_i, \mu_j) = \{l \in nm(\psi_i, \mu_j) | \text{l } is\ a\ negative\ literal\} \tag{6.21}$$

We present now the definition of the contribution to the distance coming from the non-mentioned literals. Given two clauses ψ_i and μ_j and a term similarity function $tsim : \mathcal{P} \times \mathcal{P} \rightarrow [0, 1]$, we define the distance from the non-mentioned literals as follows:

$$dnm(\psi_i, \mu_j) = \frac{1}{2}(|nnm(\psi_i, \mu_j)|) + \frac{1}{2} \sum_{t \in pnm(\psi_i, \mu_j)} (1 - mxs(letter(t), lpl(\mu_j))) \tag{6.22}$$

Negative literals in ψ_i whose associated letter is not mentioned by μ_j add 0.5 to the distance. This is because half of the models of μ_j map the letter into true and half of the models of μ_j map the letter into false. On the other hand, all the models of ψ_i map the letter into false. This leads to the expression $\frac{1}{2}(|nnm(\psi_i, \mu_j)|)$. Positive literals in ψ_i whose associated letter is not mentioned by μ_j produce an increment in distance which depends on term similarity information. Since these letters are not mentioned by μ_j, half of the models of μ_j map these letters into true and half of the models of μ_j map these letters into false. All the models of ψ_i map these letters into true. The models of μ_j mapping those letters into true add 0 to distance. Let us consider now the μ_j models that map those letters into false. There is a contradiction with the ψ_i models but, instead of adding 1 to distance, we extract the highest similarity between those differing letters and positive literals in μ_j.

Following these definitions, the distance between clauses is defined as:

$$distance(\psi_i, \mu_j) = CDistP(\psi_i, \mu_j) + CDistN(\psi_i, \mu_j) + dnm(\psi_i, \mu_j) \tag{6.23}$$

Algorithm A_TS-BRsim-1C

The simplest situation arises when both document and query are represented as conjunctions of literals. A conjunction of literals is directly in DNF and can be stored as a set with a single clause. Figure 6.5 depicts the algorithm which computes an approximation to the similarity measure TS-BRsim between a document and a query represented as conjunctions of literals. Since the return value is an approximation to TS-BRsim and the inputs are DNF formulas with a single clause, we call this algorithm A_TS-BRsim-1C.

$CDistP(\psi_1, \mu_1)$ is the number of propositional letters which appear in μ_1 as a positive literal and in ψ_1 as a negative literal. This means that all the models of the query will map these

Algorithm A_TS-BRsim-1C:

Function Similarity(ψ,μ)
Input:
query $\psi = \{\psi_1\}$
document $\mu = \{\mu_1\}$
Output:
$\approx TS - BRsim(\mu, \psi)$

1. Compute $CDistP(\psi_1, \mu_1)$
2. Compute $CDistN(\psi_1, \mu_1)$
3. Compute $dnm(\psi_1, \mu_1)$
4. $distance(\psi_1, \mu_1) = CDistP(\psi_1, \mu_1) + CDistN(\psi_1, \mu_1) + dnm(\psi_1, \mu_1)$
5. return $(1 - \frac{distance(\psi_1, \mu_1)}{|\psi_1|})$

Figure 6.5: Algorithm A_TS-BRsim-1C

letters into false and all the models of the document will map these letters into true. Recall that this sort of contradicting letters (false in the query's model and true in the document's model) increase 1 the value of the distance and, in this case, term similarity information is not used. Any model of μ_1 fares at least $CDistP(\psi_1, \mu_1)$ to the set of models of ψ_1.

The value of $CDistN(\psi_1, \mu_1)$ is computed from the set of literals $CDiffN(\psi_1, \mu_1)$. Letters belonging to $CDiffN(\psi_1, \mu_1)$ are propositional letters which appear negated in μ_1 and non-negated in ψ_1. In the following we will use the name of nc-letters (negative contradicting) to refer to the elements belonging to $CDiffN(\psi_1, \mu_1)$. All the models of the query map nc-letters into true and all the models of the document map nc-letters into false. For any model $m \in Mod(\mu_1)$, nc-letters belong to each $sd \in SSD(m, Mod(\psi_1))$. Specifically, nc-letters belong to $N(sd, m)$. Recall that letters belonging to $N(sd, m)$ increase the distance to $Mod(\psi_1)$ depending on the similarity to the terms mapped into true by m. However we are now handling document clauses and, thus, we do not know which terms are mapped into true by each model of the document. Therefore, the best we can do is to measure the contribution from nc-letters using the similarity to the terms appearing in positive literals of the clause μ_1. All the models of μ_1 map the terms appearing in positive literals into true. However some models of μ_1 will map some other letters into true (and these letters are also considered to compute the most similar term). Since we only can consider the letters appearing in a positive literal of μ_1, we are computing an approximation to the value of the distance to the set of models of the query. These considerations can be easily illustrated through an example. In fig. 6.6 we depict an example where both computations between models (left-hand side) and computations between clauses (right-hand side) are presented. The letter b, which appears in $CDiffN(\psi_1, \mu_1)$, produces a contribution to the distance which is measured using the similarity from b to the positive letters in μ_1, i.e. a, c. On the other hand, in the model-based approach the differing term b produces a contribution to the distance which depends on the similarity between the term b and the terms a, c for the model m_1, and the contribution to the distance depends on the similarity between the term b and the terms a, c, d for the model m_2. Since the term d is the most similar term to b, our approach does

$\mathcal{P}$= $\{a, b, c, d\}$
$d = a \wedge \neg b \wedge c$
$q = a \wedge b$

$tsim(b, a) = 0.1$
$tsim(b, c) = 0.6$
$tsim(b, d) = 0.7$

document models → query models↓	m_1 $\{a, c\}$	m_2 $\{a, c, d\}$
$\{a, b\}$	$\{b, c\}$	$\{b, c, d\}$
$\{a, b, c\}$	$\{b\}$	$\{b, d\}$
$\{a, b, d\}$	$\{b, c, d\}$	$\{b, c\}$
$\{a, b, c, d\}$	$\{b, d\}$	$\{b\}$

(6.6.a)Symmetric differences

$SSD(m_1, Mod(q)) = \{\{b\}\}$
$sd_{11} = \{b\}$
$P(sd_1, m_1) = \emptyset$
$N(sd_1, m_1) = \{b\}$
$mxs(b, \{a, c\}) = max\{0.1, 0.6\} = 0.6$
$Dist(m_1, Mod(q)) = 0 + (1 - 0.6) = 0.4$

$SSD(m_2, Mod(q)) = \{\{b\}\}$
$sd_{21} = \{b\}$
$P(sd_{21}, m_2) = \emptyset$
$N(sd_{21}, m_2) = \{b\}$
$mxs(b, \{a, c, d\}) = max\{0.1, 0.6, 0.7\} = 0.7$
$Dist(m_1, Mod(q)) = 0 + (1 - 0.7) = 0.3$

$distance(d, q) = \frac{0.4+0.3}{2} = 0.35$

$\mathcal{P}$= $\{a, b, c, d\}$
document $\mu = \{\mu_1\}, \mu_1 = \{a, \neg b, c\}$
query $\psi = \{\psi_1\}, \psi_1 = \{a, b\}$

$CDiff(\mu_1, \psi_1) = \{\neg b\}$
$CDiffP(\psi_1, \mu_1) = \emptyset$
$CDiffN(\psi_1, \mu_1) = \{\neg b\}$
$CDistP(\psi_1, \mu_1) = |\emptyset| = 0$
$CDistN(\psi_1, \mu_1) = (1 - mxs(b, \{a, c\})) = 1 - 0.6 = 0.4$
$nm(\psi_1, \mu_1) = \emptyset$
$pnm(\psi_1, \mu_1) = nnm(\psi_1, \mu_1) = \emptyset$
$dnm(\psi_1, \mu_1) = 0$

$distance(\psi_1, \mu_1) = 0 + 0.4 + 0 = 0.4$

Figure 6.6: Approximating TS-BRsim

not get the actual value of the distance. We take 0.4 whereas the actual value of the distance is 0.35, which is the average over the models of the document.

The contribution to the distance from query terms not mentioned by the document is measured by *dnm*. Negative query terms whose associated letter does not appear in the document's representation (terms belonging to $nnm(\psi_i, \mu_j)$, in the following nnm-terms) increase *dnm* as follows. Since half of the models of the document map nnm-terms into true and half of the models of the document map nnm-terms into false, nnm-terms only come into contradiction with half of the models of the document. Since contradicting terms which are negated by the query are directly added to the distance (no term similarity information is used), nnm-terms increase $0.5 * |nnm(\psi_i, \mu_j)|$. Let us consider now positive query terms whose associated letter does not appear in the document's representation (terms belonging to $pnm(\psi_i, \mu_j)$, in the following pnm-terms). Again, pnm-terms only come into contradiction with half of the models of the document. Nevertheless each pnm-term increases the distance depending on its similarity to terms mapped into true by the concrete document model. Since we do not know the letters mapped into true by each model of the document, we compute the similarity from each pnm-term to the terms appearing in μ_1 as positive literals. All these terms are mapped into true

by every model of the document. These considerations lead to an increment to the distance of $0.5 * \sum_{t \in pnm(\psi_1,\mu_1)}(1 - mxs(letter(t), lpl(\mu_1)))$.

An important circumstance is that if the document's clause μ_1 is a complete theory, i.e. it has a single model, the exact value of TS-BRsim is obtained through Algorithm A_TS-BRsim-1C. In this case, $nm(\psi_1, \mu_1) = \emptyset$ and, hence, $dnm(\psi_1, \mu_1) = 0$. Since μ_1 has a single model, the value $CDistP(\psi_1, \mu_1) + CDistN(\psi_1, \mu_1)$ is exactly the distance from the single model of the document to the set of models of the query.

The complexity analysis for the Algorithm A_TS-BRsim-1C is as follows. The sets $CDiffN(\psi_1, \mu_1)$ and $CDiffP(\psi_1, \mu_1)$ needed for steps 1 and 2 can be obtained in linear time with respect to the size of either ψ_1 or μ_1. $CDistP(\psi_1, \mu_1)$ is just the cardinality of $CDiffP(\psi_1, \mu_1)$ but $CDistN(\psi_1, \mu_1)$ introduces additional computations. Specifically, for each element in $CDiffN(\psi_1, \mu_1)$ we have to compute its most similar positive term in μ_1. Assuming that we can get the similarity between two terms in unit time, to process each element belonging to $CDiffN(\psi_1, \mu_1)$ we have a worst case of $|\mu_1|$ iterations. Putting all together and, taking into account that query's clauses are typically smaller than document's clauses, steps 1 and 2 can be done in $|\psi_1| * |\mu_1|$ steps. An analogous argument leads to a worst number of steps for computing $dnm(\psi_1, \mu_1)$ of $|\psi_1| * |\mu_1|$. To sum up the complexity of the algorithm is $\mathcal{O}(|\psi_1||\mu_1|)$.

Figure 6.7 presents a complete example of the execution of Algorithm A_TS-BRsim-1C. The letter a is a positive literal in both the document and the query and, thus, it does not increase the measure of distance. The letter b is negated by the document whereas it is a positive literal in the query representation. Then, b produces an increment to the distance which depends on its similarity to letters appearing in the document as positive literals ($\{a, c, f\}$). Finally, the letter e is not mentioned by the document and, hence, only half of the models of the document map it into false (contradiction with the query). This leads to an increment to the distance which is equal to $0.5 * (1 - mxs(e, \{a, c, f\}))$.

6.2.2 Distance between DNF formulas

In this section we present an algorithm that computes an approximation to the similarity measure TS-BRsim between a document and a query represented as DNF formulas with several clauses. We call this new algorithm A_TS-BRsim-SC. Basically, the policy is the same followed in section 3.1.5 to define Algorithm A_BRsim-SC.

Again, given two clauses ψ_i and μ_j, a measure of distance between them is obtained as $CDistP(\psi_i, \mu_j) + CDistN(\psi_i, \mu_j) + dnm(\psi_i, \mu_j)$.

Given μ and ψ two DNF formulas represented as $\mu = \{\mu_1, \mu_2, \dots\}$ and $\psi = \{\psi_1, \psi_2, \dots\}$, we define the distance between μ and ψ as follows.

$$\text{TS-Cdistance}(\mu, \psi) = \frac{\sum_{\mu_j \in \mu} \min_{\psi_i \in \psi} (CDistP(\psi_i, \mu_j) + CDistN(\psi_i, \mu_j) + dnm(\psi_i, \mu_j))}{|\mu|} \quad (6.24)$$

The distance between the DNF formulas is measured using the average distance from μ clauses to ψ. The distance from a μ clause to ψ is measured as the distance from the μ clause to its closest ψ clause. The distance clause-to-clause is measured as in Algorithm A_TS-BRsim-1C. Note that the measure of distance has been kept as close as possible to the semantics proposed in last sections. An average over μ clauses is done and for each μ clause the distance to ψ takes the minimum distance to ψ clauses.

$\mathcal{P} = \{a, b, c, d, e, f, g, \dots\}$

$d = a \wedge \neg b \wedge c \wedge f \wedge \neg g$
$q = a \wedge b \wedge e$

document $\mu = \{\mu_1\}, \mu_1 = \{a, \neg b, c, f, \neg g\}$
query $\psi = \{\psi_1\}, \psi_1 = \{a, b, e\}$

$tsim(b, a) = 0.1$
$tsim(b, c) = 0.2$
$tsim(b, f) = 0.4$
$tsim(e, a) = 0.1$
$tsim(e, c) = 0.3$
$tsim(e, f) = 0.1$

Algorithm A_TS-BRsim-1C:

$CDiff(\mu_1, \psi_1) = \{\neg b\}$
$CDiffP(\psi_1, \mu_1) = \emptyset$
$CDiffN(\psi_1, \mu_1) = \{\neg b\}$
Step 1: $CDistP(\psi_1, \mu_1) = |\emptyset| = 0$
$lpl(\mu_1) = \{a, c, f\}$
$mxs(b, lpl(\mu_1)) = mxs(b, \{a, c, f\}) = max\{0.1, 0.2, 0.4\} = 0.4$
Step 2: $CDistN(\psi_1, \mu_1) = (1 - mxs(b, lpl(\mu_1))) = 1 - 0.4 = 0.6$
$nm(\psi_1, \mu_1) = (\{a, b, e\} \setminus (\{a, b, e\} \cap \{a, \neg b, c, f, \neg g\})) \setminus CDiff(\psi_1, \mu_1) = (\{a, b, e\} \setminus \{a\}) \setminus \{b\} = \{e\}$
$pnm(\psi_1, \mu_1) = \{e\}$
$nnm(\psi_1, \mu_1) = \emptyset$
$dnm(\psi_1, \mu_1) = \frac{1}{2}(|nnm(\psi_1, \mu_1)|) + \frac{1}{2}\sum_{t \in pnm(\psi_1, \mu_1)}(1 - mxs(letter(t), lpl(\mu_1)))$
$dnm(\psi_1, \mu_1) = \frac{1}{2}(1 - mxs(e, lpl(\mu_1)))$
$mxs(e, \{a, c, f\}) = max\{0.1, 0.3, 0.1\} = 0.3$
Step 3: $dnm(\psi_1, \mu_1) = \frac{1}{2}(1 - mxs(e, \{a, c, f\})) = 0.5 * (1 - 0.3) = 0.35$
Step 4: $distance(\psi_1, \mu_1) = CDistP(\psi_1, \mu_1) + CDistN(\psi_1, \mu_1) + dnm(\psi_1, \mu_1) = 0 + 0.6 + 0.35 = 0.95$
Step 5: $1 - \frac{distance(\psi_1, \mu_1)}{|\psi_1|} = 1 - 0.95/3$
$return(0.683)$

Figure 6.7: Running Algorithm A_TS-BRsim-1C: an example

Algorithm A_TS-BRsim-SC:

Function Similarity(ψ,μ)
Input:
query $\psi = \{\psi_1, \psi_2, \dots\}$
document $\mu = \{\mu_1, \mu_2, \dots\}$
Output:
TS-Csim(μ, ψ)

1. $Distance = 0$;
2. Extract a new $\mu_j \in \mu$
3. $Distance\_to\_\psi = S$
4. Extract a new $\psi_i \in \psi$
5. $d = CDistN(\psi_i, \mu_j) + CDistP(\psi_i, \mu_j) + dnm(\psi_i, \mu_j)$
6. if $d < Distance\_to\_\psi$ then $Distance\_to\_\psi = d$
7. go to step 4 until no more ψ_is remain
8. $Distance+ = Distance\_to\_\psi$
9. go to step 2 until no more μ_js remain
10. $avg\_distance = \frac{Distance}{|\mu|}$
11. return$(1 - \frac{avg_distance}{\psi_{min}})$

Figure 6.8: Algorithm A_TS-BRsim-SC

The next equation transforms TS-Cdistance into a similarity measure TS-Csim in the interval [0,1]. The symbol ψ_{min} stands for the size of the smallest clause in ψ. This number is an upper bound for the distance from a μ clause to ψ.

$$\text{TS-Csim}(\mu, \psi) = 1 - \frac{\text{TS-Cdistance}(\mu, \psi)}{\psi_{min}} \tag{6.25}$$

Algorithm A_TS-BRsim-SC

The previous definitions lead directly to the Algorithm A_TS-BRsim-SC depicted in fig. 6.8. Note that Algorithm A_TS-BRsim-1C is a particular case of Algorithm A_TS-BRsim-SC. There is a number of factors why TS-Csim is an approximation to TS-BRsim. First, as argued for Algorithm A_TS-BRsim-1C, the value $CDistP(\psi_i, \mu_j) + CDistN(\psi_i, \mu_j) + dnm(\psi_i, \mu_j)$ is an approximation to the average distance from μ_j models to $Mod(\psi_i)$. Second, distinct μ clauses can have common models and, then, equation 6.24 counts common models more than once. As a result the final value of TS-Cdistance(μ, ψ) is not exactly an average over μ models.

Given two clauses ψ_i and μ_j the computation of their contribution to distance takes $|\psi_i| * |\mu_j|$ steps. Loop in step 2 takes $|\mu|$ iterations and loop in step 4 takes $|\psi|$. As a result, the complexity of the Algorithm A_TS-BRsim-SC is $\mathcal{O}(|\mu||\psi|\psi_{max}\mu_{max})$, where ψ_{max} (μ_{max}) is the size of the largest clause in ψ (μ).

$\mathcal{P} = \{a, b, c, d, e, f, g, h\}$

$d = (a \wedge \neg b \wedge c \wedge d \wedge f) \vee (a \wedge b)$
$q = (a \wedge b \wedge g) \vee (a \wedge d)$

document $\mu = \{\mu_1, \mu_2\}, \mu_1 = \{a, \neg b, c, d, f\}, \mu_2 = \{a, b\}$
query $\psi = \{\psi_1, \psi_2\}, \psi_1 = \{a, b, g\}, \psi_2 = \{a, d\}$

$tsim(b, a) = 0.1$
$tsim(b, c) = 0.2$
$tsim(b, d) = 0.5$
$tsim(b, f) = 0.4$
$tsim(g, a) = 0.1$
$tsim(g, b) = 0.2$
$tsim(g, c) = 0.3$
$tsim(g, d) = 0.2$
$tsim(g, f) = 0.1$
$tsim(d, a) = 0.2$

Algorithm A_TS-BRsim-SC:

1. $Distance = 0$
2. $\mu_1 = \{a, \neg b, c, d, f\}$
   3. $Distance\_to\_\psi = 8$
   4. $\psi_1 = \{a, b, g\}$
      $CDiff(\mu_1, \psi_1) = \{\neg b\}$
      $CDiffP(\psi_1, \mu_1) = \emptyset$
      $CDiffN(\psi_1, \mu_1) = \{\neg b\}$
      $CDistP(\psi_1, \mu_1) = |\emptyset| = 0$
      $lpl(\mu_1) = \{a, c, d, f\}$
      $mxs(b, lpl(\mu_1)) = mxs(b, \{a, c, d, f\}) = max\{0.1, 0.2, 0.5, 0.4\} = 0.5$
      $CDistN(\psi_1, \mu_1) = (1 - mxs(b, lpl(\mu_1))) = 1 - 0.5 = 0.5$
      $nm(\psi_1, \mu_1) = (\{a, b, g\} \setminus (\{a, b, g\} \cap \{a, \neg b, c, d, f\})) \setminus CDiff(\psi_1, \mu_1) = (\{a, b, g\} \setminus \{a\}) \setminus \{b\} = \{g\}$
      $pnm(\psi_1, \mu_1) = \{g\}$
      $nnm(\psi_1, \mu_1) = \emptyset$
      $dnm(\psi_1, \mu_1) = \frac{1}{2}(|nnm(\psi_1, \mu_1)|) + \frac{1}{2}\sum_{l \in pnm(\psi_1, \mu_1)}(1 - mxs(letter(l), lpl(\mu_1)))$
      $dnm(\psi_1, \mu_1) = \frac{1}{2}(1 - mxs(g, lpl(\mu_1)))$
      $mxs(g, \{a, c, d, f\}) = max\{0.1, 0.3, 0.2, 0.1\} = 0.3$
      $dnm(\psi_1, \mu_1) = 0.5 * (1 - 0.3) = 0.35$
      5. $d = CDistP(\psi_1, \mu_1) + CDistN(\psi_1, \mu_1) + dnm(\psi_1, \mu_1) = 0 + 0.5 + 0.35 = 0.85$
      6. $Distance\_to\_\psi = 0.85$
   4. $\psi_2 = \{a, d\}$
      $CDiff(\mu_1, \psi_2) = \emptyset$
      $CDiffP(\psi_2, \mu_1) = \emptyset$
      $CDiffN(\psi_2, \mu_1) = \emptyset$
      $CDistP(\psi_2, \mu_1) = |\emptyset| = 0$
      $CDistN(\psi_2, \mu_1) = |\emptyset| = 0$
      $nm(\psi_2, \mu_1) = (\{a, d\} \setminus (\{a, d\} \cap \{a, \neg b, c, d, f\})) \setminus CDiff(\psi_2, \mu_1) = (\{a, d\} \setminus \{a, d\}) \setminus \emptyset = \emptyset$
      $pnm(\psi_2, \mu_1) = \emptyset$
      $nnm(\psi_2, \mu_1) = \emptyset$
      $dnm(\psi_2, \mu_1) = 0$
      5. $d = CDistP(\psi_2, \mu_1) + CDistN(\psi_2, \mu_1) + dnm(\psi_2, \mu_1) = 0$
      6. $Distance\_to\_\psi = 0$
   8. $Distance = 0$

(continued)

Figure 6.9: Running Algorithm A_TS-BRsim-SC: an example (I)

Algorithm A_TS-BRsim-SC:

2. $\mu_2 = \{a, b\}$
   3. $Distance\_to\_\psi = 8$
   4. $\psi_1 = \{a, b, g\}$
      $CDiff(\mu_2, \psi_1) = \emptyset$
      $CDiffP(\psi_1, \mu_2) = \emptyset$
      $CDiffN(\psi_1, \mu_2) = \emptyset$
      $CDistP(\psi_1, \mu_2) = |\emptyset| = 0$
      $CDistN(\psi_1, \mu_2) = 0$
      $nm(\psi_1, \mu_2) = (\{a, b, g\} \setminus (\{a, b, g\} \cap \{a, b\})) \setminus CDiff(\psi_1, \mu_2) = (\{a, b, g\} \setminus \{a, b\}) \setminus \emptyset = \{g\}$
      $pnm(\psi_1, \mu_2) = \{g\}$
      $nnm(\psi_1, \mu_2) = \emptyset$
      $dnm(\psi_1, \mu_2) = \frac{1}{2}(|nnm(\psi_1, \mu_2)|) + \frac{1}{2}\sum_{l \in pnm(\psi_1, \mu_2)}(1 - mxs(letter(l), lpl(\mu_2)))$
      $dnm(\psi_1, \mu_2) = \frac{1}{2}(1 - mxs(g, lpl(\mu_2)))$
      $mxs(g, \{a, b\}) = max\{0.1, 0.2\} = 0.2$
      $dnm(\psi_1, \mu_2) = 0.5 * (1 - 0.2) = 0.4$
      5. $d = CDistP(\psi_1, \mu_2) + CDistN(\psi_1, \mu_2) + dnm(\psi_1, \mu_2) = 0 + 0 + 0.4 = 0.4$
      6. $Distance\_to\_\psi = 0.4$
   4. $\psi_2 = \{a, d\}$
      $CDiff(\mu_2, \psi_2) = \emptyset$
      $CDiffP(\psi_2, \mu_2) = \emptyset$
      $CDiffN(\psi_2, \mu_2) = \emptyset$
      $CDistP(\psi_2, \mu_2) = |\emptyset| = 0$
      $CDistN(\psi_2, \mu_2) = 0$
      $nm(\psi_2, \mu_2) = (\{a, d\} \setminus (\{a, d\} \cap \{a, b\})) \setminus CDiff(\psi_2, \mu_2) = (\{a, d\} \setminus \{a\}) \setminus \emptyset = \{d\}$
      $pnm(\psi_2, \mu_2) = \{d\}$
      $nnm(\psi_2, \mu_2) = \emptyset$
      $dnm(\psi_2, \mu_2) = \frac{1}{2}(|nnm(\psi_2, \mu_2)|) + \frac{1}{2}\sum_{l \in pnm(\psi_2, \mu_2)}(1 - mxs(letter(l), lpl(\mu_2)))$
      $dnm(\psi_2, \mu_2) = \frac{1}{2}(1 - mxs(d, lpl(\mu_2)))$
      $mxs(d, \{a, b\}) = max\{0.2, 0.5\} = 0.5$
      $dnm(\psi_2, \mu_2) = 0.5 * (1 - 0.5) = 0.25$
      5. $d = CDistP(\psi_2, \mu_2) + CDistN(\psi_2, \mu_2) + dnm(\psi_2, \mu_2) = 0.25$
      6. $Distance\_to\_\psi = 0.25$
   8. $Distance = 0.25$
10. $avg\_distance = \frac{Distance}{|\mu|} = \frac{0.25}{2} = 0.125$
$1 - \frac{avg\_distance}{\psi_{min}} = 1 - \frac{0.125}{2} = 0.9375$
11. return(0.9375)

Figure 6.10: Running Algorithm A_TS-BRsim-SC: an example (II)

Figures 6.9 and 6.10 present an example of the execution of Algorithm A_TS-BRsim-SC. Again, some values of *tsim* are assumed. The distance from the document clause $\{a, \neg b, c, d, f\}$ to the query is 0 because it satisfies all the terms of the query clause $\{a, d\}$. The distance from the document clause $\{a, b\}$ to the query is given by the distance from $\{a, b\}$ to the query clause $\{a, d\}$.

6.3 Incorporating inverse document frequency

In this section we extend the model to deal with inverse document frequency. We introduce a new measure of distance from a given interpretation I to the set of models of a formula ψ. First, we define the *matching letters* between I and $Mod(\psi)$. Then, we propose that matching letters produce an increment in the distance from I to $Mod(\psi)$ which depends on the idf of the terms involved.

First, we collect all the differing letters between I and models of ψ.

$$dl(I, Mod(\psi)) = \{\cup sd | sd = I \triangle J, J \in Mod(\psi)\}, \tag{6.26}$$

That is, $dl(I, Mod(\psi))$ contains the propositional letters with a differing interpretation between I and at least one interpretation $J \in Mod(\psi)$. On the other hand, all the letters not belonging to $dl(I, Mod(\psi))$ are mapped into the same truth value by I and all the models of ψ. Then, we define the matching letters between I and $Mod(\psi)$ as follows.

$$ml(I, Mod(\psi)) = \mathcal{P} \setminus dl(I, Mod(\psi)) \tag{6.27}$$

Letters belonging to $ml(I, Mod(\psi))$ are mapped into the same truth value by I and all the models of ψ. We can divide the set of matching letters into two complementary sets.

$$pml(I, Mod(\psi)) = \{l \in ml(I, Mod(\psi)) | l \in I\} \tag{6.28}$$

$$nml(I, Mod(\psi)) = \{l \in ml(I, Mod(\psi)) | l \notin I\} \tag{6.29}$$

The model presented in the previous section considers that matching letters increase 0 the value of the distance from I to the set of models of ψ.

Matching letters belonging to $pml(I, Mod(\psi))$ are mapped into true by I and all the models of ψ. For instance, this corresponds with a document mentioning the matching letters and a query asking for those letters. We propose that these matching letters produce an increment in the distance which depends on idf information. Basically, not all these matching letters are equally good. Depending on their significance within the whole collection (idf), they will produce a small or a big increment on the distance. This connects directly with IR matching functions in which a matching term with high idf often produces an increment in the similarity which is bigger than the one obtained from a matching term with low idf. On the other hand, matching letters belonging to $nml(I, Mod(\psi))$ are mapped into false by I and all the models of ψ. For instance, this corresponds with a document that does not deal with the concepts represented by the matching letters and a query asking for documents that do not mention those letters. In this case we think that the use of idf makes not sense. As a result, we propose to define the distance from an interpretation I to the set of models of a formula ψ as follows.

$$Dist(I, Mod(\psi)) = \min_{sd \in SSD(I, Mod(\psi))} (|P(sd, I)| + \sum_{l \in N(sd,I)} (1 - mxs(l, I))) + \tag{6.30}$$

$$+\alpha * \sum_{l \in pml(I, Mod(\psi))} (1 - idf(l)),$$

where the function $idf : \mathcal{P} \rightarrow [0, 1]$ is a normalized version of the inverse document frequency and α is a tuning value in the interval [0,1] measuring the importance of the idf factor. A matching letter l such that $idf(l) \approx 1$ is a good matching term (the term appears in few documents) and, thus, the increment in the distance from this letter is nearly 0. On the other hand, a matching letter l such that $idf(l) \approx 0$ is a bad term (it appears in many documents) and, thus, the increment in the distance is nearly α. Clearly, a value of $\alpha = 1$ is not fair. Consider a matching letter l such that $idf(l) = 0$. If $\alpha = 1$ this letter would increase 1 the value of distance. This would not be appropriate, since non-matching letters increase distance at most 1. Although the letter l is not a very significant term (because $idf(l) = 0$), it keeps being a matching term. Hence, matching letters should produce smaller increments in distance, i.e. $\alpha < 1$.

As before, given a formula ψ, a total preorder between interpretations can be extracted from the nearness of each interpretation to the set of models of the formula ψ. Formally, a total preorder $\leq_\psi$ is defined as:

$$I \leq_\psi J \quad \text{iff} \quad Dist(I, Mod(\psi)) \leq Dist(J, Mod(\psi)) \tag{6.31}$$

Now, the notion of closeness takes into account term similarity information and inverse document frequency. Again, we can define $<$ as $I < J$ if and only if $I \leq J$ and $J \not\leq I$.

Following this order we can define a new revision operator. When revising a theory ψ with a new information μ, the models of the new information which are the closest to the theory (w.r.t the new order induced by the theory) are selected to be the models of the revised theory:

$$Mod(\psi \circ \mu) = Min(Mod(\mu), \leq_\psi) \tag{6.32}$$

It can be easily demonstrated that the new assignment is not faithful because, given two models of ψ, I and I', it can be the case that $I <_\psi I'$ because of the distance introduced by the matching terms.

Again, the distance between a document and a query is given by the average distance from models of the document to the set of models of the query:

$$distance(d, q) = \frac{\sum_{m \in Mod(d)} Dist(m, Mod(q))}{|Mod(d)|} \tag{6.33}$$

A total preorder $\leq_q$ induced by the query can be defined. This preorder establishes formally a method to rank documents in terms of their respective distances to the query. Now this order takes into account both term similarity information and inverse document frequency.

$$d_i \leq_q d_j \quad \text{iff} \quad distance(d_i, q) \leq distance(d_j, q) \tag{6.34}$$

We can also define $<_q$ as $d_i <_q d_j$ iff $d_i \leq_q d_j$ and not $d_j \leq_q d_i$, and $=_q$ as $d_i =_q d_j$ iff $d_i \leq_q d_j$ and $d_j \leq_q d_i$.

A new similarity measure in the interval [0,1] can be defined from the previous definitions. This measure is called IDF-TS-BRsim.

$$\text{IDF-TS-BRsim}(d, q) = 1 - distance(d, q)/k, \tag{6.35}$$

where k is the number of propositional letters appearing in q. Figure 6.11 shows an example of the computation of IDF-TS-BRsim. Some values of $tsim$ and idf are assumed. The tuning value α is assumed to be equal to 0.5. Note that the matching letter a produces an increment to the distance which depends on the value of $idf(a)$.

6.4 Implementation

In this section, we present efficient algorithms that calculate an approximate value to the similarity measure IDF-TS-BRsim. These algorithms are slight variations of Algorithms A_TS-BRsim-1C and A_TS-BRsim-SC to deal with the notion of idf. First, we refine the measure of distance between clauses defined in section 6.2.1. Given two clauses ψ_i and μ_j, $distance(\psi_i, \mu_j)$ computes an approximate value to the average distance from models of μ_j to the set of models of ψ_i. Since now the distance from a model of μ_j to the set of models of ψ_i considers idf information, we propose a number of definitions to determine the contribution of matching letters to this distance. Later on, in the next section, we present the algorithm that computes an approximate value to IDF-TS-BRsim(μ, ψ), where μ and ψ are DNF formulas with one clause representing a document and a query respectively. This algorithm is based on the computation of the distance between the query and the document clause as defined here.

Given two clauses ψ_i and μ_j the next formula collects the common literals between the clauses:

$$CL(\psi_i, \mu_j) = \psi_i \cap \mu_j \tag{6.36}$$

Common literals can be positive or negative. Since we are only interested in common positive literals, we collect them into CPL as follows:

$$CPL(\psi_i, \mu_j) = \{l \in CL(\psi_i, \mu_j) | l \text{ is a positive literal}\} \tag{6.37}$$

Given two clauses ψ_i and μ_j and a function $idf : \mathcal{P} \rightarrow [0, 1]$ the next formula defines the contribution to the distance coming from positive matching literals.

$$dm(\psi_i, \mu_j) = \alpha * \sum_{l \in CPL(\psi_i, \mu_j)} (1 - idf(l)) \tag{6.38}$$

Following these definitions, the distance between two clauses ψ_i and μ_j is defined as:

$$distance(\psi_i, \mu_j) = CDistP(\psi_i, \mu_j) + CDistN(\psi_i, \mu_j) + dnm(\psi_i, \mu_j) + dm(\psi_i, \mu_j) \tag{6.39}$$

We present now the Algorithms A_IDF-TS-BRsim-1C and A_IDF-TS-BRsim-SC that compute an approximate value to the similarity measure IDF-TS-BRsim.

$\mathcal{P}= \{a, b, c, d\}$
$d = a \wedge b \wedge \neg c$
$q = a \wedge c$

$tsim(c, a) = 0.1$
$tsim(c, b) = 0.3$
$tsim(c, d) = 0.4$

$idf(a) = 0.5$

document models → / query models↓	m_1 $\{a, b\}$	m_2 $\{a, b, d\}$
$\{a, c\}$	$\{b, c\}$	$\{b, c, d\}$
$\{a, b, c\}$	$\{c\}$	$\{c, d\}$
$\{a, c, d\}$	$\{b, c, d\}$	$\{b, c\}$
$\{a, b, c, d\}$	$\{c, d\}$	$\{c\}$

Symmetric differences between query and document models

$SSD(m_1, Mod(q)) = \{\{c\}\}$
$sd_{11} = \{c\}$
$P(sd_{11}, m_1) = \emptyset$
$N(sd_{11}, m_1) = \{c\}$
$dl(m_1, Mod(q)) = \{b, c, d\}$
$ml(m_1, Mod(q)) = \{a, b, c, d\} \setminus \{b, c, d\} = \{a\}$
$pml(m_1, Mod(q)) = \{a\}$
$nml(m_1, Mod(q)) = \emptyset$
$Dist(m_1, Mod(q)) = (1 - mxs(c, m_1)) + \alpha * (1 - idf(a))$
$mxs(c, m_1) = mxs(c, \{a, b\}) = max\{0.1, 0.3\} = 0.3$
$Dist(m_1, Mod(q)) = (1 - 0.3) + 0.5 * (1 - 0.5) = (1 - 0.3) + 0.25 = 0.95$

$SSD(m_2, Mod(q)) = \{\{c\}\}$
$sd_{21} = \{c\}$
$P(sd_{21}, m_2) = \emptyset$
$N(sd_{21}, m_2) = \{c\}$
$dl(m_2, Mod(q)) = \{b, c, d\}$
$ml(m_2, Mod(q)) = \{a, b, c, d\} \setminus \{b, c, d\} = \{a\}$
$pml(m_2, Mod(q)) = \{a\}$
$nml(m_2, Mod(q)) = \emptyset$
$Dist(m_2, Mod(q)) = 0 + (1 - mxs(c, m_2)) + \alpha * (1 - idf(a))$
$mxs(c, m_2) = mxs(c, \{a, b, d\}) = max\{0.1, 0.3, 0.4\} = 0.4$
$Dist(m_2, Mod(q)) = (1 - 0.4) + 0.5 * (1 - 0.5) = 0.85$

$distance(d, q) = \frac{0.95+0.85}{2} = 0.8$
IDF-TS-BRsim$(d, q) = 1 - 0.8/2 = 0.6$

Figure 6.11: An example of the computation of IDF-TS-BRsim

6.4.1 Algorithm A_IDF-TS-BRsim-1C

Given a document μ and a query ψ represented as DNF formulas with a single clause, Algorithm A_IDF-TS-BRsim-1C computes an approximate value to IDF-TS-BRsim(μ, ψ). Algorithm A_IDF-TS-BRsim-1C, depicted in fig.6.12, computes the contribution to distance from non-matching terms as Algorithm A_TS-BRsim-1C does. Besides, the computation of the contribution to distance from matching terms is introduced.

The set $CPL(\psi_1, \mu_1)$ contains the common positive literals between the document's clause and the query's clause. It can be easily proved that, for any model I of the document it holds that $CPL(\psi_1, \mu_1) \subseteq pml(I, Mod(\psi))$. Let us consider a letter $l \in CPL(\psi_1, \mu_1)$. This means that all the models of the document and all the models of the query have to map l into true. Hence, no one symmetric difference can contain l. Consequently, for any model I of the document $l \in pml(I, Mod(\psi))$. On the other hand, if a letter l belongs to $pml(I, Mod(\psi))$ it is not ensured that l belongs to $CPL(\psi_1, \mu_1)$ because there can be other models of the document that map l into false and, hence, l would not belong to $CPL(\psi_1, \mu_1)$. Since we only use $CPL(\psi_1, \mu_1)$ we are computing an approximate value to the actual contribution to distance from matching terms. This is because we compute the average contribution to the distance using the matching terms that are common to all the models of the document ($CPL(\psi_1, \mu_1)$).

Algorithm A_IDF-TS-BRsim-1C:

Function Similarity(ψ,μ)
Input:
query $\psi = \{\psi_1\}$
document $\mu = \{\mu_1\}$
Output:
$\approx$ IDF-TS-BRsim(μ, ψ)

1. Compute $CDistP(\psi_1, \mu_1)$
2. Compute $CDistN(\psi_1, \mu_1)$
3. Compute $dnm(\psi_1, \mu_1)$
4. Compute $CPL(\psi_1, \mu_1)$
5. Compute $dm(\psi_1, \mu_1)$
6. $distance(\psi_1, \mu_1) = CDistP(\psi_1, \mu_1) + CDistN(\psi_1, \mu_1) + dnm(\psi_1, \mu_1) + dm(\psi_1, \mu_1)$
7. return $(1 - \frac{distance(\psi_1, \mu_1)}{|\psi_1|})$

Figure 6.12: Algorithm A_IDF-TS-BRsim-1C

The additional steps introduced do not increase the level of the complexity of the algorithm with regard to Algorithm A_TS-BRsim-1C. This is because $CPL(\psi_1, \mu_1)$ can be computed in linear time with respect to the size of any of the involved clauses. Figure 6.13 presents a complete example of the execution of Algorithm A_IDF-TS-BRsim-1C. A tuning value of $\alpha = 0.5$ is assumed. Note that the common positive literals, $\{a, b\}$ produce an increment to the distance which depends on the values $idf(a)$ and $idf(b)$.

$\mathcal{P} = \{a, b, c, d, e, f, g, \dots\}$

$d = a \wedge b \wedge c \wedge f \wedge \neg g$
$q = a \wedge b \wedge \neg c$

document $\mu = \{\mu_1\}, \mu_1 = \{a, b, c, f, \neg g\}$
query $\psi = \{\psi_1\}, \psi_1 = \{a, b, \neg c\}$

$idf(a) = 0.6$
$idf(b) = 0.8$

Algorithm A_IDF-TS-BRsim-1C:

$CDiff(\mu_1, \psi_1) = \{c\}$
$CDiffP(\psi_1, \mu_1) = \{c\}$
$CDiffN(\psi_1, \mu_1) = \emptyset$
Step 1: $CDistP(\psi_1, \mu_1) = |\{c\}| = 1$
Step 2: $CDistN(\psi_1, \mu_1) = 0$
$nm(\psi_1, \mu_1) = (\{a, b, \neg c\} \setminus (\{a, b, \neg c\} \cap \{a, b, c, f, \neg g\})) \setminus CDiff(\psi_1, \mu_1)$
$nm(\psi_1, \mu_1) = (\{a, b, \neg c\} \setminus \{a, b\}) \setminus \{\neg c\} = \emptyset$
$pnm(\psi_1, \mu_1) = \emptyset$
$nnm(\psi_1, \mu_1) = \emptyset$
Step 3: $dnm(\psi_1, \mu_1) = 0$
$CL(\psi_1, \mu_1) = \psi_1 \cap \mu_1 = \{a, b\}$
Step 4: $CPL(\psi_1, \mu_1) = \{a, b\}$
Step 5: $dm(\psi_1, \mu_1) = \alpha * ((1 - idf(a)) + (1 - idf(b))) = 0.5 * ((1 - 0.6) + (1 - 0.8)) = 0.3$
Step 6: $distance(\psi_1, \mu_1) = CDistP(\psi_1, \mu_1) + CDistN(\psi_1, \mu_1) + dnm(\psi_1, \mu_1) + dm(\psi_1, \mu_1)$
$distance(\psi_1, \mu_1) = 1 + 0 + 0 + 0.3 = 1.3$
Step 7: $1 - \frac{distance(\psi_1, \mu_1)}{|\psi_1|} = 1 - 1.3/3$
$return(0.567)$

Figure 6.13: Running Algorithm A_IDF-TS-BRsim-1C: an example

6.4.2 Algorithm A_IDF-TS-BRsim-SC

Now we follow the same policy that was applied to design Algorithm A_TS-BRsim-SC from the definition of the measure of distance between clauses used in Algorithm A_TS-BRsim-1C. We can define a measure of distance between generic DNF formulas using the measure of distance between clauses used in the previous section.

Formally, given μ and ψ two DNF formulas represented as $\mu = \{\mu_1, \mu_2, \dots\}$ and $\psi = \{\psi_1, \psi_2, \dots\}$, we define the distance between μ and ψ as follows.

$$\text{IDF-TS-Cdistance}(\mu, \psi) = \frac{\sum_{\mu_j \in \mu} \min_{\psi_i \in \psi} (distance(\psi_i, \mu_j))}{|\mu|} \tag{6.40}$$

This is an approximate value to $distance(\mu, \psi)$ (defined in equation 6.33). Again, common models between clauses are counted more than once and, besides, the measure of distance between clauses is also an approximation, as argued in the previous section.

The next equation transforms IDF-TS-Cdistance into a similarity measure IDF-TS-Csim in the interval [0,1]. The symbol ψ_{min} stands for the size of the smallest clause in ψ, which is an upper bound for the distance from a μ clause to ψ.

$$\text{IDF-TS-Csim}(\mu, \psi) = 1 - \frac{\text{IDF-TS-Cdistance}(\mu, \psi)}{\psi_{min}} \tag{6.41}$$

Algorithm A_IDF-TS-BRsim-SC, depicted in fig. 6.14, is a straightforward modification of Algorithm A_TS-BRsim-SC that includes the idf factor. Figures 6.15 and 6.16 present an example of the execution of Algorithm A_IDF-TS-BRsim-SC. Some values of *idf* are assumed. The tuning value α is assumed to be equal to 0.5. Now, clauses are matched using term similarity and idf information. Again, for each document clause, its closest query clause determines the distance from the document clause to the query.

6.5 Evaluation

We carried out additional experiments to evaluate the model presented in this chapter. In order to test the effect of term similarity we need a measure of similarity between terms. The problem of defining a measure of similarity between terms has been addressed by many researchers. We have used the *Expected Mutual Information Measure (EMIM)* because it has been used with success in the past [11, 16]. Given two terms t_i and t_j, $EMIM(t_i, t_j)$ is often interpreted as a measure of the statistical information contained in t_i about t_j (or vice versa, it being a symmetric measure). Formally, the EMIM measure is defined as follows:

$$EMIM(t_i, t_j) = \sum_{t_i, t_j} P(t_i \in d, t_j \in d) log \frac{P(t_i \in d, t_j \in d)}{P(t_i \in d) P(t_j \in d)} \tag{6.42}$$

where t_i and t_j are any two terms of the term space T. Van Rijsbergen [58] proposed a method to estimate EMIM between two terms using co-occurrence data that can be derived by a statistical analysis of the term occurrences in the collection.

Algorithm A_IDF-TS-BRsim-SC:

Function Similarity(ψ,μ)
Input:
query $\psi = \{\psi_1, \psi_2, \dots\}$
document $\mu = \{\mu_1, \mu_2, \dots\}$
Output:
IDF-TS-Csim(μ, ψ)

1. $Distance = 0$;
   2. Extract a new $\mu_j \in \mu$
   3. $Distance\_to\_\psi = S$
      4. Extract a new $\psi_i \in \psi$
      5. $d = CDistN(\psi_i, \mu_j) + CDistP(\psi_i, \mu_j) + dnm(\psi_i, \mu_j) + dm(\psi_i, \mu_j)$
      6. if $d < Distance\_to\_\psi$ then $Distance\_to\_\psi = d$
      7. go to step 4 until no more ψ_is remain
   8. $Distance+ = Distance\_to\_\psi$
   9. go to step 2 until no more μ_js remain
10. $avg\_distance = \frac{Distance}{|\mu|}$
11. return$(1 - \frac{avg_distance}{\psi_{min}})$

Figure 6.14: Algorithm A_IDF-TS-BRsim-SC

$\mathcal{P} = \{a, b, c, d, e, f, g, h\}$

$d = (a \wedge b \wedge c \wedge d \wedge f) \vee (a \wedge b \wedge c)$
$q = (a \wedge c) \vee (a \wedge b)$

document $\mu = \{\mu_1, \mu_2\}, \mu_1 = \{a, b, c, d, f\}, \mu_2 = \{a, b, c\}$
query $\psi = \{\psi_1, \psi_2\}, \psi_1 = \{a, c\}, \psi_2 = \{a, b\}$

$idf(a) = 0.7$
$idf(b) = 0.5$
$idf(c) = 0.9$

Algorithm A_IDF-TS-BRsim-SC:

1. $Distance = 0$
2. $\mu_1 = \{a, b, c, d, f\}$
   3. $Distance\_to\_\psi = 8$
   4. $\psi_1 = \{a, c\}$
      $CDiff(\mu_1, \psi_1) = \emptyset$
      $CDiffP(\psi_1, \mu_1) = \emptyset$
      $CDiffN(\psi_1, \mu_1) = \emptyset$
      $CDistP(\psi_1, \mu_1) = |\emptyset| = 0$
      $CDistN(\psi_1, \mu_1) = 0$
      $nm(\psi_1, \mu_1) = (\{a, c\} \setminus (\{a, c\} \cap \{a, b, c, d, f\})) \setminus CDiff(\psi_1, \mu_1) = (\{a, c\} \setminus \{a, c\}) \setminus \emptyset = \emptyset$
      $pnm(\psi_1, \mu_1) = \emptyset$
      $nnm(\psi_1, \mu_1) = \emptyset$
      $dnm(\psi_1, \mu_1) = 0$
      $CL(\psi_1, \mu_1) = \psi_1 \cap \mu_1 = \{a, c\}$
      $CPL(\psi_1, \mu_1) = \{a, c\}$
      $dm(\psi_1, \mu_1) = \alpha * ((1 - idf(a)) + (1 - idf(c))) = 0.5 * ((1 - 0.7) + (1 - 0.9)) = 0.2$
      5. $d = CDistP(\psi_1, \mu_1) + CDistN(\psi_1, \mu_1) + dnm(\psi_1, \mu_1) + dm(\psi_1, \mu_1) = 0.2$
      6. $Distance\_to\_\psi = 0.2$
   4. $\psi_2 = \{a, b\}$
      $CDiff(\mu_1, \psi_2) = \emptyset$
      $CDiffP(\psi_2, \mu_1) = \emptyset$
      $CDiffN(\psi_2, \mu_1) = \emptyset$
      $CDistP(\psi_2, \mu_1) = |\emptyset| = 0$
      $CDistN(\psi_2, \mu_1) = 0$
      $nm(\psi_2, \mu_1) = (\{a, b\} \setminus (\{a, b\} \cap \{a, b, c, d, f\})) \setminus CDiff(\psi_2, \mu_1) = (\{a, b\} \setminus \{a, b\}) \setminus \emptyset = \emptyset$
      $pnm(\psi_2, \mu_1) = \emptyset$
      $nnm(\psi_2, \mu_1) = \emptyset$
      $dnm(\psi_2, \mu_1) = 0$
      $CL(\psi_2, \mu_1) = \psi_2 \cap \mu_1 = \{a, b\}$
      $CPL(\psi_2, \mu_1) = \{a, b\}$
      $dm(\psi_2, \mu_1) = \alpha * ((1 - idf(a)) + (1 - idf(b))) = 0.5 * ((1 - 0.7) + (1 - 0.5)) = 0.4$
      5. $d = CDistP(\psi_2, \mu_1) + CDistN(\psi_2, \mu_1) + dnm(\psi_2, \mu_1) + dm(\psi_2, \mu_1) = 0.4$
   8. $Distance = 0 + 0.2 = 0.2$

(continued)

Figure 6.15: Running Algorithm A_IDF-TS-BRsim-SC: an example (I)

Algorithm A_IDF-TS-BRsim-SC:

2. $\mu_2 = \{a, b, c\}$
   3. $Distance\_to\_\psi = 8$
   4. $\psi_1 = \{a, c\}$
      $CDiff(\mu_2, \psi_1) = \emptyset$
      $CDiffP(\psi_1, \mu_2) = \emptyset$
      $CDiffN(\psi_1, \mu_2) = \emptyset$
      $CDistP(\psi_1, \mu_2) = |\emptyset| = 0$
      $CDistN(\psi_1, \mu_2) = 0$
      $nm(\psi_1, \mu_2) = (\{a, c\} \setminus (\{a, c\} \cap \{a, b, c\})) \setminus CDiff(\psi_1, \mu_2) = (\{a, c\} \setminus \{a, c\}) \setminus \emptyset = \emptyset$
      $pnm(\psi_1, \mu_2) = \emptyset$
      $nnm(\psi_1, \mu_2) = \emptyset$
      $dnm(\psi_1, \mu_2) = 0$
      $CL(\psi_1, \mu_2) = \psi_1 \cap \mu_2 = \{a, c\}$
      $CPL(\psi_1, \mu_2) = \{a, c\}$
      $dm(\psi_1, \mu_2) = \alpha * ((1 - idf(a)) + (1 - idf(c))) = 0.5 * ((1 - 0.7) + (1 - 0.9)) = 0.2$
      5. $d = CDistP(\psi_1, \mu_2) + CDistN(\psi_1, \mu_2) + dnm(\psi_1, \mu_2) + dm(\psi_1, \mu_2) = 0.2$
      6. $Distance\_to\_\psi = 0.2$
   4. $\psi_2 = \{a, b\}$
      $CDiff(\mu_2, \psi_2) = \emptyset$
      $CDiffP(\psi_2, \mu_2) = \emptyset$
      $CDiffN(\psi_2, \mu_2) = \emptyset$
      $CDistP(\psi_2, \mu_2) = |\emptyset| = 0$
      $CDistN(\psi_2, \mu_2) = 0$
      $nm(\psi_2, \mu_2) = (\{a, b\} \setminus (\{a, b\} \cap \{a, b, c\})) \setminus CDiff(\psi_2, \mu_2) = (\{a, b\} \setminus \{a, b\}) \setminus \emptyset = \emptyset$
      $pnm(\psi_2, \mu_2) = \emptyset$
      $nnm(\psi_2, \mu_2) = \emptyset$
      $dnm(\psi_2, \mu_2) = 0$
      $CL(\psi_2, \mu_2) = \psi_2 \cap \mu_2 = \{a, b\}$
      $CPL(\psi_2, \mu_2) = \{a, b\}$
      $dm(\psi_2, \mu_2) = \alpha * ((1 - idf(a)) + (1 - idf(b))) = 0.5 * ((1 - 0.7) + (1 - 0.5)) = 0.4$
      5. $d = CDistP(\psi_2, \mu_2) + CDistN(\psi_2, \mu_2) + dnm(\psi_2, \mu_2) + dm(\psi_2, \mu_2) = 0.4$
   8. $Distance = 0.2 + 0.2 = 0.4$
10. $avg\_distance = \frac{Distance}{|\mu|} = \frac{0.4}{2} = 0.2$
$1 - \frac{avg\_distance}{\psi_{min}} = 1 - \frac{0.2}{2} = 0.9$
11. return(0.9)

Figure 6.16: Running Algorithm A_IDF-TS-BRsim-SC: an example (II)

We have used the EMIM data for the CACM collection[1]. Since the EMIM data file is extremely big, we had to apply some restrictions in order to decrease the computational effort. Given a term t only its most similar terms are considered. Specifically, we cut off the similarity values at three decimals. This means that similarity values below 0.001 are considered as 0. Recall that we use the measure of term similarity to compute the similarity between a given term t to its most similar term in a set of terms A. Then, we inspect the EMIM data file looking for the list of terms similar to t (as argued before this list of terms only contains terms whose similarity to t is at least 0.001) and we determine which term belonging to A presents the highest similarity to t. In the pathological case that arises when all the terms in the set A do not appear in the list of terms similar to t, we just take a similarity value of 0.

Since the EMIM data file stores the terms stemmed by the Porter stemmer, we also used this stemmer in the experiments presented here. We implemented the algorithms A_TS-BRsim-SC and A_IDF-TS-BRsim-SC and we used them to compute similarity between documents and queries. Again, we ran AND-tests, which stored representations as DNF formulas with a single clause (only conjunctions), and AND/OR tests, which stored representations as generic DNF formulas.

First, we present the experiments that incorporated the term similarity measure. Specifically, we used the algorithm A_TS-BRsim-SC to compute an approximate value to the measure TS-BRsim. Then, we show the results of the experiments that incorporated both term similarity and inverse document frequency. In this case we used the algorithm A_IDF-TS-BRsim-SC to compute an approximate value to the measure IDF-TS-BRsim. All the experiments have been conducted against the CACM collection.

6.5.1 Term Similarity

Table 6.1 summarizes the results obtained after applying term similarity in the model. Figure 6.17 presents the same results in a graphical way. The first two columns of the table show the results for AND-tests and the last columns show the results obtained for AND/OR-tests. The baselines were obtained as follows. For the AND-tests, we ran a test in the same conditions than the best AND-test presented in chapter 3 but we used the Porter stemmer instead of the SMART-enhanced version of the Lovins stemmer. An analogous procedure was applied to get the baseline for the AND/OR-tests. This means that baseline tests applied Algorithm A_BRsim-SC to compute similarity and, then, we can compare the model without term similarity ($BRsim$ measure) against the new model incorporating term similarity.

Significant improvements are obtained when term similarity is introduced. The use of term similarity information leads to better performance results for both cases, with representations as DNF clauses with a single clause and with representations as DNF clauses with several clauses.

6.5.2 Term similarity and inverse document frequency

In order to evaluate the model with the idf factor we computed the inverse document frequency of each stemmed term t, i.e. we calculated $log\frac{N}{n_t}$, where N is the size of the collection and n_t is the number of documents in which the term t appears. We normalized these values in the interval [0,1] by dividing by its maximum value, i.e. $logN$. These computations allowed us to define the function idf, which is required by the algorithm A_IDF-TS-BRsim-SC. We tried out

[1]We thank Dr. Fabio Crestani for providing us with the EMIM data for the CACM collection.

Recall - Precision	AND-test	AND-test With term similarity	AND/OR-test	AND/OR-test With term similarity
0.00	0.5805	0.6074	0.6105	0.6342
0.10	0.4340	0.4762	0.5077	0.5300
0.20	0.3384	0.3476	0.3960	0.4324
0.30	0.2436	0.2946	0.3205	0.3518
0.40	0.1929	0.2384	0.2709	0.2928
0.50	0.1608	0.1824	0.2449	0.2670
0.60	0.1273	0.1477	0.1809	0.1821
0.70	0.0817	0.0818	0.1111	0.1190
0.80	0.0698	0.0642	0.0967	0.1007
0.90	0.0427	0.0401	0.0535	0.0570
1.00	0.0342	0.0319	0.0402	0.0448
Avg. prec.	0.2096	0.2284	0.2575	0.2738
% Prec. change		+9.0%		+6.3%
Avg. prec. for 3 intermediate points	0.1897	0.1980	0.2459	0.2667
% Prec. change		+4.4%		+8.5%

Table 6.1: Term similarity

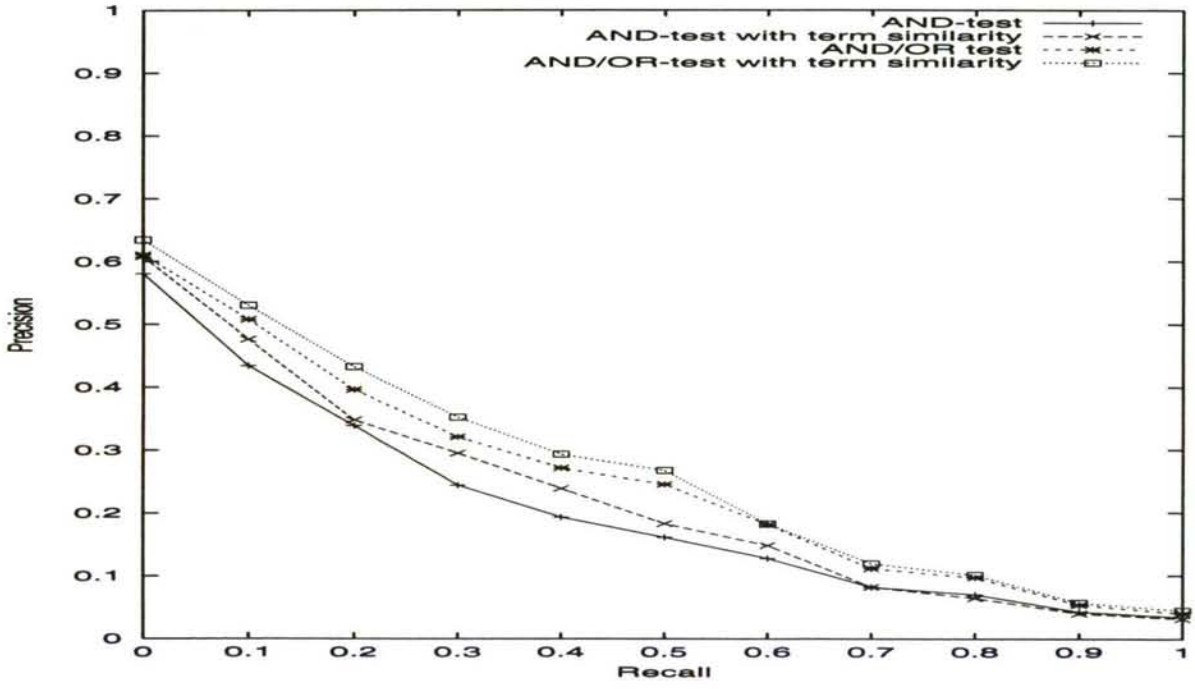

Figure 6.17: Term similarity

values for the tuning constant α (which determines the importance of the idf factor) from 0.9 to 0.1 in steps of 0.1 and we show here the best results.

Table 6.2 and figure 6.18 summarize the results obtained. Spectacular improvements were achieved after the introduction of the idf factor. The introduction of term similarity and the inverse document factor results in very significant improvements of the performance ratios for both AND-tests and AND/OR-tests.

Indeed the results obtained for the AND/OR-test with term similarity and inverse document frequency are comparable to the results obtained from classical models using non-binary weights. In this respect, we used SMART to produce performance results for the vector space model with the tf/idf weighting scheme (for those who are familiar with SMART, we used the `ntn` weighting scheme). Table 6.3 (and fig. 6.19) shows our best run against the tf/idf vector-space run in SMART. This does not pretend to be a strict comparison since the vector-space model does not apply term similarity for the non-matching terms and, on the other hand, the PLBR model cannot cope with term frequencies. Nevertheless this comparison illustrates the close difference between a classical model and our model after the introduction of term similarity and inverse document frequency.

The evaluation of the extension of the model presented in this chapter is really encouraging. Although our model keeps dealing with binary-weighted terms (we represent documents and queries as propositional formulas), the introduction of term similarity and the idf factor in the computation of the similarity between a document and a query has led to spectacular improvements in performance. As a matter of fact, when we represent documents and queries as generic DNF formulas the extension of the model produces performance results which are comparable to classical models handling tf/idf weights. This shows that using an expressive framework the performance is not so dependent on the term frequency factor. Moreover, to avoid the tf factor reduces the computational cost of the system.

The proposal presented here is especially promising for future extensions of the model that apply more ellaborated techniques to get generic DNF formulas. Note that the results presented here correspond to DNF formulas whose clauses are simply constructed from the subfields of the documents/queries. On the other hand, methods such as Passage retrieval techniques that divide documents and queries in a more precise way seem to be suitable tools to get further improvements in the performance of the system. In this respect, a future line of work is the evaluation of the model against bigger collections that provide us big documents and, hence, ellaborated divisions into several clauses can be articulated. TREC or its subcollections are good candidates for future experiments.

Recall - Precision	AND-test	AND-test With ts and idf Best $\alpha = 0.6$	AND/OR-test	AND/OR-test With ts and idf Best $\alpha = 0.5$
0.00	0.5805	0.6228	0.6105	0.7568
0.10	0.4340	0.4970	0.5077	0.6218
0.20	0.3384	0.3849	0.3960	0.5279
0.30	0.2436	0.3251	0.3205	0.4284
0.40	0.1929	0.2866	0.2709	0.3906
0.50	0.1608	0.2449	0.2449	0.3353
0.60	0.1273	0.2105	0.1809	0.2754
0.70	0.0817	0.1274	0.1111	0.1919
0.80	0.0698	0.1132	0.0967	0.1590
0.90	0.0427	0.0826	0.0535	0.1012
1.00	0.0342	0.0703	0.0402	0.0835
Avg. prec.	0.2096	0.2696	0.2575	0.3520
% Prec. change		+28.6%		+36.7%
Avg. prec. for 3 intermediate points	0.1897	0.2477	0.2459	0.3407
% Prec. change		+30.6%		+38.6%

Table 6.2: Term similarity and inverse document frequency

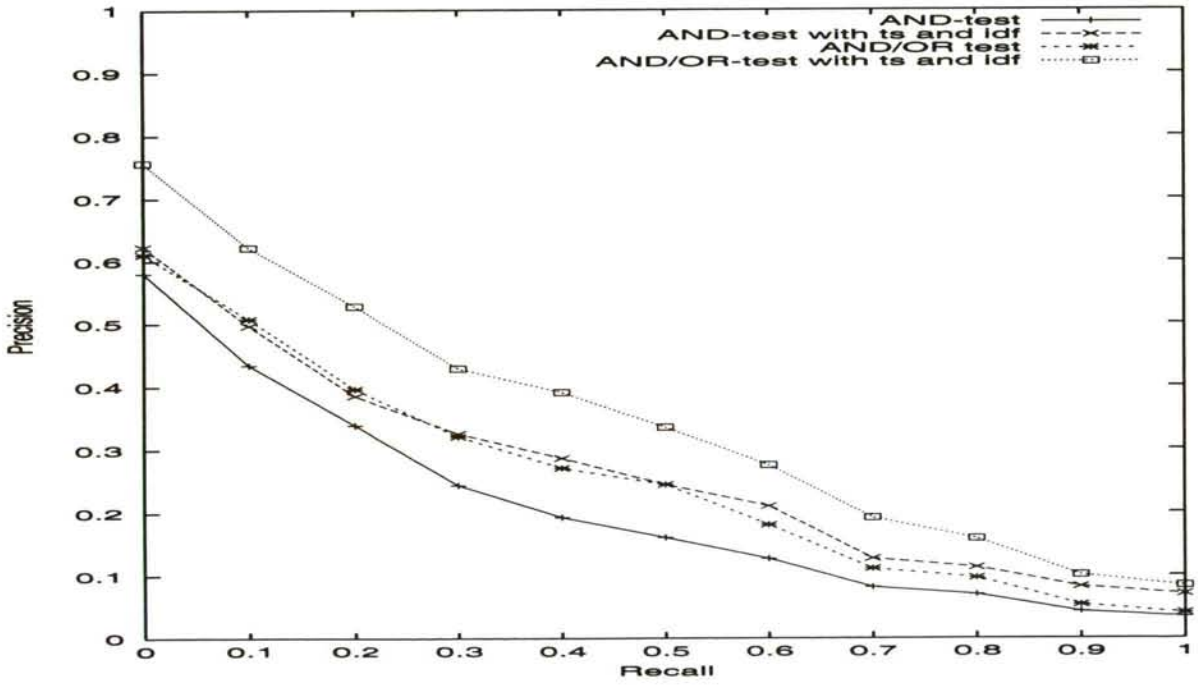

Figure 6.18: Term similarity and inverse document frequency

Recall - Precision	PLBR model AND/OR-test with ts and idf	Vector-space tf/idf
0.00	0.7568	0.7366
0.10	0.6218	0.6377
0.20	0.5279	0.5236
0.30	0.4284	0.4522
0.40	0.3906	0.4027
0.50	0.3353	0.3493
0.60	0.2754	0.2918
0.70	0.1919	0.2188
0.80	0.1590	0.1915
0.90	0.1012	0.1406
1.00	0.0835	0.1202
Avg. prec. % Prec. change	0.3520	0.3695 +5%
Avg. prec. for 3 intermediate points % Prec. change	0.3407	0.3548 +4.1%

Table 6.3: AND/OR test with ts and idf vs. classical tf/idf

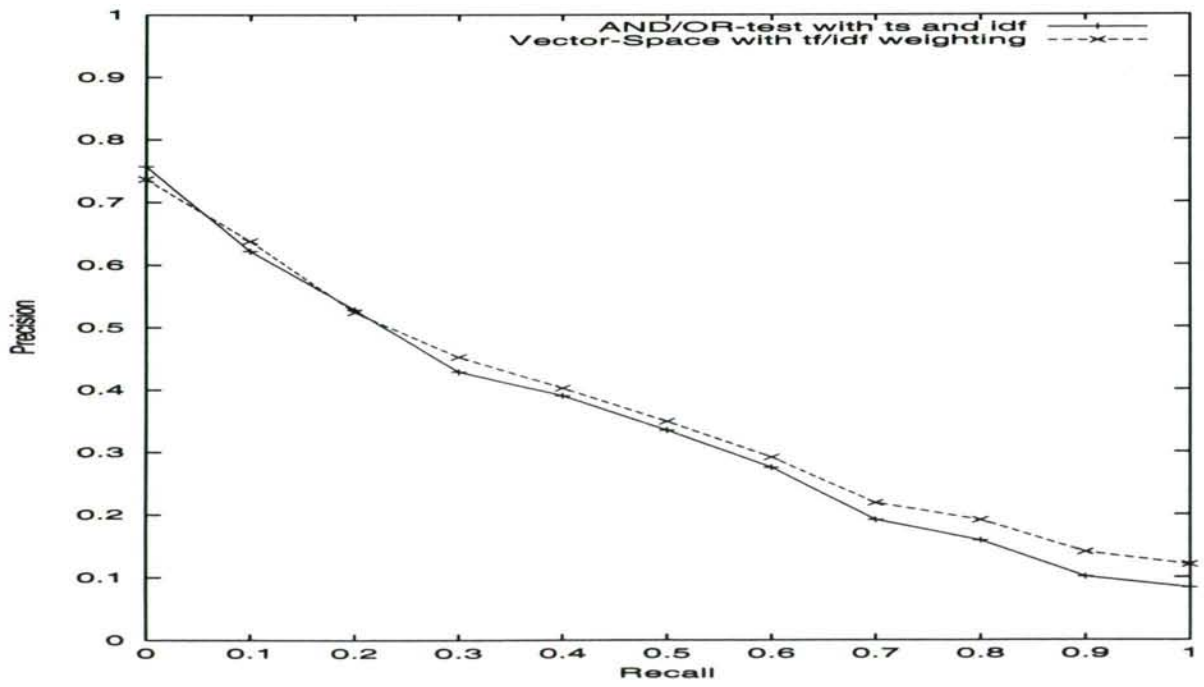

Figure 6.19: AND/OR test with ts and idf vs. classical tf/idf

Conclusions

In this thesis we followed a logical approach to model IR. From a basic framework we instantiated some classical IR tasks, studied carefully their computational costs and proposed efficient implementations. In all the stages we stressed the advantages of the logical approach. In this respect, we showed that the logical model is general since its representational capabilities are very flexible. Furthermore, we used the same logical framework, Propositional Logic, to model distinct IR elements. This leads to an homogeneous framework in which distinct notions are encompassed.

The first part of this thesis was dedicated to model the basic retrieval task. Since the use of the notion of logical consequence is too coarse, we took van Rijsbergen's logical uncertainty principle as our basis to define a non-binary measure of relevance. The field of Belief Revision has thoroughly studied measures of closeness between logical models. We took advantage from these works to formalize a measure of relevance between documents and queries represented as logical formulas. Within this basic model we are able to represent binary-weighted classical vectors and, besides, the measure of relevance corresponds to the inner product query-document matching function. However, Propositional Logic is inherently more expressive because it can deal with more ellaborated representations. Several views of documents and queries can be expressed. Negations can effectively be used to express the actual contents of a document and the actual intentions of a user. This leads to representations of documents and queries which are more accurate. Image retrieval and speech retrieval can also benefit from the enhanced expressiveness supplied by logic.

Computational properties are especially critical in IR systems. Then, we studied in depth the complexity of the tasks involved in the model. As a result, we proposed efficient algorithms able to compute similarity between documents and queries in polynomial time. Once the model was efficiently implemented, we evaluated it against standard IR collections. Indeed, few logical models have been evaluated. Since the formalism underlying our proposal is simple enough, we were able to extract representations for documents and queries in an automatic way. Furthermore, the structure of subfields of these collections allowed us to design experiments to test the impact of enhanced representations on retrieval performance. The results of these experiments were really promising. First, the actual applicability of the model was ensured. In fact, a restricted case of the model performs roughly the same as the classical vector-space model with binary weights. Second, the use of more expressive representations results in significant improvements in retrieval performance. This shows that an expressive formalism is desirable. Besides, the improvements in performance were obtained using a simple technique for dividing documents into clauses. The application of more ellaborated techniques - such as Passage retrieval ones - to divide documents into clauses is a potential tool to get increasingly better performance results.

We also modeled retrieval situations and relevance feedback within our model. Retrieval situations have been proposed in the literature to encompass several elements that make influence on

a relevance judgment. We have proposed the inclusion of retrieval situations through a revision process. The representations of a retrieval situation and a document are combined and, hence, a better relevance judgment can be made. Besides, we presented an efficient algorithm that implements the revision process between a retrieval situation and a document. The relevance feedback process was also formalized through a revision process. Basically, the initial representation of a query is revised with feedback information. The task of feedback makes evident some advantages of the logical approach. Since the formalism is general, we were able to expand the query with both negative and positive terms. The use of negative terms is an important precision-oriented mechanism because it allows to move the query away from unrelevant documents. This is a lack in classical feedback approaches. Indeed, the evaluation we carried out has clearly shown that the use of negative terms results in significant improvements in retrieval performance.

The last chapter of this thesis is focused on the extension of the model to deal with term similarity and inverse document frequency. We defined new measures of similarity between documents and queries. The formalism to represent documents and queries keeps being Propositional Logic but these new factors are now taken into account by the measure of similarity. Since the extension of the model is homogeneous, we could inherit the good behavior of the basic model without term similarity and inverse document frequency. New algorithms were developed and new experiments were run. The retrieval performance results are very encouraging for the future of this research. As a matter of fact, the performance of our model is close to the performance of classical models using non-binary weighting schemes.

We think that the future of the model presented here is promising. We have modeled a number of tasks in which the use of expressive representations led to better retrieval performance results. Since the tradeoff between expressiveness and complexity was carefully studied, the model is open to be effectively applied in conjunction with classical methods to extract representations for documents and queries. Furthermore, more ellaborated methods that capture the knowledge implicit in documents and queries can benefit from the framework proposed here. Indeed, we strongly believe that the articulation of expressive representations which are close to the actual semantics of the document/query is the first step to solve the IR problem.

Appendix A
Tests of statistical significance

PLBRM AND-test (TWKA) vs the best PLBRM AND/OR-test (TWKA) in CISI

In the CISI collection, the best PLBRM AND/OR-test (only ANDs in document's representations & ANDs/ORs in representations of queries) performed better than the best PLBRM AND-test. However, since the difference is not too large (AND-test average precision: 0.1487, AND/OR-test average precision: 0.1598), we decided to run statistical significance tests in order to determine whether or not this difference is statistically significant.

For each query, we compute the value of the variable:

D_i = avg. prec. of query Q_i for AND-test − avg. prec. of query Q_i for AND/OR-test

T-Test

We have 76 variables D_i that we use to compute the value of the variable T:

$$\overline{D} = \sum_{i=1}^{n} D_i = -0.8462$$

$$T = \frac{\overline{D}}{\frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(D_i-\overline{D})^2}}{\sqrt{n}}} = -8.7526$$

Finally, the value of T is used to compute the p-value:

$$\text{p-value} = 1.0 - F(n-1, |T|) = 1.0 - F(75, 8.7526)$$

, where F is the cumulative distribution function of the t-distribution with 75 degrees of freedom. Looking at the table of this distribution function we can observe that $F(75, 8.7526) \approx 1$, then p-value ≈ 0. Then, following the t-test, we can conclude that the difference between the best PLBRM AND/OR-test and the best PLBRM AND-test is statistically significant.

PLBRM AND-test (TW) vs PLBRM AND/OR-test (TW) in LISA

In the LISA collection, the PLBRM AND/OR-test performed better than the best PLBRM AND-test. However, since the difference is not too large (AND-test average precision: 0.0600, AND/OR-test average precision: 0.0647), we decided to run statistical significance tests in order to determine whether or not this difference is statistically significant.

For each query, we compute the value of the variable:

D_i = avg. prec. of query Q_i for AND-test − avg. prec. of query Q_i for AND/OR-test

T-Test

We have 35 variables D_i that we use to compute the value of the variable T:

$$\overline{D} = \sum_{i=1}^{n} D_i = -0.1071$$

$$T = \frac{\overline{D}}{\frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(D_i-\overline{D})^2}}{\sqrt{n}}} = -3.4745$$

Finally, the value of T is used to compute the p-value:

$$\text{p-value} = 1.0 - F(n-1, |T|) = 1.0 - F(34, 3.4745)$$

, where F is the cumulative distribution function of the t-distribution with 34 degrees of freedom. Looking at the table of this distribution function we can observe that $F(34, 3.4745) \approx 1$, then p-value ≈ 0. Then, following the t-test, we can conclude that the difference between the best PLBRM AND/OR-test and the best PLBRM AND-test is statistically significant.

Base residual vs document oriented feedback in CISI

The CISI collection presented a small improvement when applying document oriented feedback. In terms of average precision, the run that used document oriented feedback performed 5.2% better than the base residual run. As the difference is not too large, we show here the results of the statistical significance tests.

For each query, we compute the value of the variable:

D_i = avg. prec. of query Q_i for base residual − avg. prec. of query Q_i for doc. oriented feedback

T-Test

We have 60 variables D_i that we use to compute the value of the variable T:

$$\overline{D} = \sum_{i=1}^{n} D_i = -0.4059$$

$$T = \frac{\overline{D}}{\frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(D_i - \overline{D})^2}}{\sqrt{n}}} = -7.7798$$

Finally, the value of T is used to compute the p-value:

$$\text{p-value} = 1.0 - F(n-1, |T|) = 1.0 - F(59, 7.7798)$$

, where F is the cumulative distribution function of the t-distribution with 59 degrees of freedom. Looking at the table of this distribution function we can observe that $F(59, 7.7798) \approx 1$, then p-value ≈ 0. Then, following the t-test, we can conclude that the difference between the base residual test and the test that applied document oriented feedback is statistically significant.

Appendix B
List of common words

Our prototype system used the following list of common words in the stopword phase. The list is the same used by the SMART system.

List of common words

a a's able about above according accordingly across actually after afterwards again against ain't all allow allows almost alone along already also although always am among amongst an and another any anybody anyhow anyone anything anyway anyways anywhere apart appear appreciate appropriate are aren't around as aside ask asking associated at available away awfully b be became because become becomes becoming been before beforehand behind being believe below beside besides best better between beyond both brief but by c c'mon c's came can can't cannot cant cause causes certain certainly changes clearly co com come comes concerning consequently consider considering contain containing contains corresponding could couldn't course currently d definitely described despite did didn't different do does doesn't doing don't done down downwards during e each edu eg eight either else elsewhere enough entirely especially et etc even ever every everybody everyone everything everywhere ex exactly example except f far few fifth first five followed following follows for former formerly forth four from further furthermore g get gets getting given gives go goes going gone got gotten greetings h had hadn't happens hardly has hasn't have haven't having he he's hello help hence her here here's hereafter hereby herein hereupon hers herself hi him himself his hither hopefully how howbeit however i i'd i'll i'm i've ie if ignored immediate in inasmuch inc indeed indicate indicated indicates inner insofar instead into inward is isn't it it'd it'll it's its itself j just k keep keeps kept know knows known l last lately later latter latterly least less lest let let's like liked likely little look looking looks ltd m mainly many may maybe me mean meanwhile merely might more moreover most mostly much must my myself n name namely nd near nearly necessary need needs neither never nevertheless new next nine no nobody non none noone nor normally not nothing novel now nowhere o obviously of off often oh ok okay old on once one ones only onto or other others otherwise ought our ours ourselves out outside over overall own p particular particularly per perhaps placed please plus possible presumably probably provides q que quite qv r rather rd re really reasonably regarding regardless regards relatively respectively right s said same saw say saying says second secondly see seeing seem seemed seeming seems seen self selves sensible sent serious seriously seven several shall she should shouldn't since six so some somebody somehow someone something sometime sometimes somewhat somewhere soon sorry specified specify specifying still sub such sup sure t t's take taken tell tends th than thank thanks thanx that that's thats the their theirs them

themselves then thence there there's thereafter thereby therefore therein theres thereupon these they they'd they'll they're they've think third this thorough thoroughly those though three through throughout thru thus to together too took toward towards tried tries truly try trying twice two u un under unfortunately unless unlikely until unto up upon us use used useful uses using usually uucp v value various very via viz vs w want wants was wasn't way we we'd we'll we're we've welcome well went were weren't what what's whatever when whence whenever where where's whereafter whereas whereby wherein whereupon wherever whether which while whither who who's whoever whole whom whose why will willing wish with within without won't wonder would would wouldn't x y yes yet you you'd you'll you're you've your yours yourself yourselves z zero

Bibliography

[1] C. Alchourrón and D. Makinson. On the logic of theory change: contraction functions and their associated revision functions. *Theoria*, 48:14–37, 1982.

[2] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, ACM press, 1999.

[4] C. L. Barry. User-defined relevance criteria:an exploratory study. *Journal of the American Society of Information Science*, 45:149–159, 1994.

[5] A. Borgida. Language features for flexible handling of exceptions in information systems. *ACM Transactions on Database Systems*, 10:563–603, 1985.

[6] P. Bruza and B. van Linder. Preferential models of query by navigation. In F. Crestani, M. Lalmas, and C. J. Van Rijsbergen, editors, *Information Retrieval, Uncertainty and Logics:advanced models for the representation and retrieval of information*, pages 73–96, Norwell, MA, 1998. Kluwer Academic Publishers.

[7] P. D. Bruza and M. Lalmas. Logic-based information retrieval: is it really worth it? In *Proc. EUFIT'96, the 4th European Congress on Intelligent Techniques and Soft Computing*, pages 841–845, Aachen, Germany, 1997.

[8] C. Buckley and G. Salton. Optimization of relevance feedback weights. In *Proc. of SIGIR-95, the 18th ACM Conference on Research and Development in Information Retrieval*, pages 351–357, Seattle, USA, July 1995.

[9] J.P. Callan. Passage-level evidence in document retrieval. In *Proc. SIGIR-94, the 17th ACM Conference on Research and Development in Information Retrieval*, pages 302–310, Dublin, UK, July 1994.

[10] J-P. Chevallet and Y. Chiaramella. Experiences in information retrieval modelling using structured formalisms and modal logic. In M. Lalmas F. Crestani and C. J. van Rijsbergen, editors, *Information Retrieval, Uncertainty and Logics: advanced models for the representation and retrieval of information*, pages 39–72, Norwell, MA, 1998. Kluwer Academic.

[11] F. Crestani. Exploiting the similarity of non-matching terms at retrieval time. *Information Retrieval*, 2(1):25–45, 2000.

[12] F. Crestani, M. Lalmas, and C. J. van Rijsbergen (editors). *Information Retrieval, Uncertainty and Logics: advanced models for the representation and retrieval of information.* Kluwer Academic, Norwell, MA., 1998.

[13] F. Crestani, I. Ruthven, M. Sanderson, and C. J. van Rijsbergen. The troubles with using a logical model of information retrieval on a large collection of documents. experimenting retrieval by logical imaging. In *Proc. TREC-4, the Fourth Text Retrieval Conference*, pages 509–525, Washington, USA, 1995.

[14] F. Crestani and C. J. van Rijsbergen. Information retrieval by logical imaging. *Journal of Documentation*, 51(1):1–15, 1995.

[15] F. Crestani and C. J. van Rijsbergen. Probability kinematics in information retrieval. In *Proc. of SIGIR-95, the 18th ACM Conference on Research and Development in Information Retrieval*, pages 291–299, Seattle, USA, July 1995.

[16] F. Crestani and C. J. van Rijsbergen. A study of probability kinematics in information retrieval. *ACM Transactions on Information Systems*, 16(3):225–255, 1998.

[17] M. Dalal. Investigations into a theory of knowledge base revision:preliminary report. In *Proc. AAAI-88, the 7th National Conference on Artificial Intelligence*, pages 475–479, Saint Paul, USA, 1988.

[18] A. del Val. *Belief Revision and Update.* PhD thesis, Stanford University, 1993.

[19] A. del Val. Syntactic characterizations of belief change operators. In *Proc. IJCAI'93, the Thirteenth International Joint Conference on Artificial Intelligence*, pages 540–545, Chambery, France, 1993.

[20] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.

[21] R. Fagin, J.D. Ullman, and M.Y. Vardi. On the semantics of updates in databases. In *Proc. of the 2nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Databases Systems*, pages 352–365, 1983.

[22] P. Gärdenfors. Rules for rational changes of belief. In T.Pauli, editor, *Philosophical Essays Dedicated to Lennart Aqvist on His Fiftieth Birthday*, pages 88–101, 1982.

[23] P. Gärdenfors. Belief revisions and the ramsey test for conditionals. *The Philosophical Review*, 95(1):81–93, 1986.

[24] D. Harman. Towards interactive query expansion. In *Proc. SIGIR-88, the 11th ACM International Conference on Research and Development in Information Retrieval*, pages 321–331, Grenoble, France, June 1988.

[25] D. Harman. Relevance feedback revisited. In *Proc. SIGIR-92, the 15th ACM International Conference on Research and Development in Information Retrieval*, pages 1–10, Copenhagen, Denmark, June 1992.

[26] D. Harper and C. J. van Rijsbergen. An evaluation of feedback in document retrieval using co-occurrence data. *Journal of Documentation*, 34(3):189–216, 1978.

[27] M. Hearst and C. Plaunt. Subtopic structuring for full-length document access. In *Proc. SIGIR-93, the 16th ACM Conference on Research and Development in Information Retrieval*, pages 59–68, Pittsburgh, USA, June 1993.

[28] K. Hoashi, K. Matsumoto, N. Inoue, and K. Hashimoto. Document filtering method using non-relevant information profile. In *Proc. SIGIR-2000, the 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 176–183, Athens, Greece, july 2000.

[29] E. Ide. New experiments in relevance feedback. In G. Salton, editor, *The SMART retrieval system - Experiments in automatic document processing*, pages 337–354, Englewood Cliffs, NJ, 1971. Prentice Hall Inc.

[30] H. Katsuno and A. O. Mendelzon. On the difference between updating a knowledge base and revising it. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proc. of KR'91, the 2nd International Conference on Principles on Knowledge Representation and Reasoning*, pages 387–393, Cambridge, USA, 1991.

[31] H. Katsuno and A. O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.

[32] H. Katsuno and A. O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.

[33] T. Kindo, H. Yoshida, T. Morimoto, and T. Watanabe. Adaptative personal information filtering system that organizes personal profiles automatically. In *Proc. IJCAI'97, the 15th International Conference on Artificial Intelligence*, pages 716–721, Nagoya, Japan, 1997.

[34] M. Lalmas and P. D. Bruza. The use of logic in information retrieval modeling. *Knowledge Engineering Review*, 13(3):263–295, 1998.

[35] M. Lalmas and C. J. van Rijsbergen. A logical model of information retrieval based on situation theory. In *Proc. of the 14th British Computer Society Information Retrieval Colloquium*, pages 1–13, Lancaster, UK, 1992.

[36] R. Lau, A. H. M. ter Hofstede, and P. D. Bruza. Applying maxi-adjustment to adaptative information filtering agents. In *Proc. NMR'2000, the 8th International Workshop on Non-Monotonic Reasoning*, Beckenridge, USA, 2000.

[37] J. H. Lee. Properties of extended boolean models in information retrieval. In *Proc. of SIGIR-94, the 17th ACM Conference on Research and Development in Information Retrieval*, Dublin, Ireland, July 1994.

[38] D. Lewis. Probability of conditionals and conditionals probabilities. In W.L. Harper, R. Stalnaker, and G. Pearce, editors, *The University of Western Ontario Series in Philosophy of Science*, pages 129–147, Dordrecht, Holland, 1981. D.Reidel Publishing Company.

[39] D. E. Losada and A. Barreiro. Using a belief revision operator for document ranking in extended boolean models. In *Proc. SIGIR-99, the 22nd ACM Conference on Research and Development in Information Retrieval*, pages 66–73, Berkeley, USA, August 1999.

[40] D. E. Losada and A. Barreiro. Efficient algorithms for ranking documents represented as dnf formulas. In *Proc. SIGIR-2000 Workshop on Mathematical and Formal Methods in Information Retrieval*, pages 16–24, Athens, Greece, July 2000.

[41] D. E. Losada and A. Barreiro. Implementing document ranking within a logical framework. In *Proc. SPIRE-2000, the 7th Symposium on String Processing and Information Retrieval*, pages 188–198, A Coruña, Spain, September 2000.

[42] D. E. Losada and A. Barreiro. Retrieval situations and belief change. In *Proc. LUMIS'2000, the DEXA'2000 International Workshop on Logical and Uncertainty Models for Information Systems*, pages 531–537, Greenwich, UK, September 2000.

[43] D. E. Losada and A. Barreiro. An homogeneous framework to model relevance feedback. In *Poster session of SIGIR-2001, the 24th ACM Conference on Research and Development in Information Retrieval (to appear)*, New Orleans, USA, September 2001.

[44] J. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.

[45] C. Meghini, F. Sebastiani, U. Straccia, and C. Thanos. A model of information retrieval based on a terminological logic. In *Proc. of SIGIR-93, the 16th ACM Conference on Research and Development in Information Retrieval*, pages 298–307, Pittsburgh, PA, June 1993.

[46] J-Y. Nie. An outline of a general model for information retrieval systems. In *Proc. SIGIR-88, the 11th ACM Conference on Research and Development in Information Retrieval*, pages 495–506, Grenoble, France, June 1988.

[47] J-Y. Nie. An information retrieval model based on modal logic. *Information Processing and Management*, 25(5):477–491, 1989.

[48] J-Y. Nie. Towards a probabilistic modal logic for semantic-based information retrieval. In *Proc. of SIGIR-92, the 15th ACM Conference on Research and Development in Information Retrieval*, pages 140–151, Copenhague, Denmark, June 1992.

[49] J-Y. Nie, M. Brisebois, and F. Lepage. Information retrieval as counterfactual. *The Computer Journal*, 38(8):643–657, 1995.

[50] F.P. Ramsey. General propositions and causality. In R.B. Braithwaite, editor, *Foundations of Mathematics and Other Logical Essays*, pages 237–257, New York, 1950. Routledge and Kegan Paul.

[51] J.J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART retrieval system - Experiments in automatic document processing*, Englewood Cliffs, NJ, 1971. Prentice Hall Inc.

[52] G. Salton. Evaluation problems in interactive information retrieval. *Information Storage and Retrieval*, 6(1):29–44, 1970.

[53] G. Salton. *The SMART Retrieval System - Experiments in Automatic Document Processing.* Prentice Hall, Englewood Cliffs, NJ, 1971.

[54] G. Salton, E. A. Fox, and H. Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(12):1022–1036, 1983.

[55] K. Satoh. Nonmonotonic reasoning by minimal belief revision. In *Proc. of the International Conference on Fifth Generation Computer Systems*, pages 455–462, Tokyo, Japan, 1988.

[56] F. Sebastiani. On the role of logic in information retrieval. *Information Processing and Management*, 34(1):1–18, 1998.

[57] F. Sebastiani. Towards a logical reconstruction of information retrieval theory. *Cybernetics and Systems*, 30(5):411–428, 1999.

[58] C. J. van Rijsbergen. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation*, 33(2):106–119, 1977.

[59] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, 1979.

[60] C. J. van Rijsbergen. Towards an information logic. In *Proc. of SIGIR-89, the 12th ACM Conference on Research and Development in Information Retrieval*, pages 77–86, Cambridge, MA, June 1989.

[61] C.J. van Rijsbergen. A non-classical logic for information retrieval. *The Computer Journal*, 29:481–485, 1986.

[62] A. Weber. Updating propositional formulas. In *Proc. of the 1st Conference on Expert Database Systems*, pages 487–500, Tokyo, Japan, 1986.

[63] M-A. Williams. Iterated theory base change: A computational model. In C. S. Mellish, editor, *Proc. of IJCAI'95, the 14th International Joint Conference on Artificial Intelligence*, pages 1541–1547, San Francisco, USA, 1995.

[64] M-A. Williams. Implementing belief revision. In G. Antoniou, editor, *Nonmonotonic Reasoning*, pages 197–211, Cambridge, USA, 1997. MIT press.

[65] M. Winslett. Reasoning about action using a possible models approach. In *Proc. of AAAI-88, the 7th National Conference on Artificial Intelligence*, pages 89–93, St. Paul, USA, 1988.